POLITECNICO DI TORINO

Corso di Laurea Magistrale in Ingegneria Informatica

Tesi di Laurea Magistrale

Risparmio Energetico in Reti TSCH Multi-Hop

Politecnico di Torino

Relatori

Dr. Stefano Scanzio

Dr. Gianluca Cena

Candidato

Federico Quarta

Dicembre 2021

Alla mia famiglia,
per avermi sempre incoraggiato e supportato

"Tutto ciò che puoi immaginare è reale"
Pablo Picasso

Sommario

Uno dei requisiti principali delle reti di sensori wireless (WSN) è il risparmio energetico. La modalità operativa TSCH dello standard IEEE 802.15.4, grazie al meccanismo di accesso time slotted, permette di ridurre consistentemente i consumi energetici, dando ai dispositivi la possibilità di mantenere l'interfaccia di comunicazione spenta e di riattivarla solo nei timeslot riservati alla trasmissione o ricezione di frame di dati. Tuttavia, una considerevole quantità di energia viene comunque sprecata a causa dell'idle listening. Questo fenomeno si verifica ogni volta che il dispositivo attiva il modulo wireless per ascoltare il canale in attesa di un frame dati, ma non viene effettuata nessuna trasmissione.
PRIL è una delle tecniche sviluppate per limitare lo spreco di energia causato dall'idle listening. La sua prima implementazione, PRIL-F, permette di ridurre i consumi sui nodi della rete che si trovano sul primo hop e, pertanto, è in grado di offrire ottimi risultati solo per reti con topologia a stella o con alcune caratteristiche specifiche.
Il lavoro svolto in questa tesi consiste nella progettazione, e successivo sviluppo, di una tecnica PRIL in grado di superare i limiti di PRIL-F. In particolare, è stata ideata la tecnica PRIL-MH, che offre un meccanismo di riduzione della quantità di energia sprecata a causa del fenomeno di idle listening in reti di sensori wireless multi-livello. La versione PRIL-MHB rappresenta l'implementazione base della tecnica e fornisce ottimi risultati in termini di energia risparmiata, ma ha lo svantaggio di peggiorare le latenze di comunicazione. Per tale motivo, a partire da PRIL-MHB, sono state sviluppate due ulteriori versioni di PRIL-MH (PRIL-MHI e PRIL-MHI2) che offrono dei meccanismi in grado di limitare l'aumento delle latenze, pur mantenendo i consumi ridotti. I risultati ottenuti dalle simulazioni in topologie multi-livello mostrano che, in condizioni tipiche, le tecniche PRIL-MH sono molto efficaci e permettono di ridurre l'energia totale sprecata della rete a causa del fenomeno di idle listening fino a tre ordini di grandezza rispetto a PRIL-F.

Indice

Elenco delle figure

Elenco delle tabelle

Listati

Capitolo 1

Introduzione

Negli ultimi anni, il progresso nei campi della microelettronica e delle tecnologie radio a basso consumo ha portato alla creazione di sensori wireless di piccole dimensioni caratterizzati da un costo di produzione basso e da un consumo energetico ridotto. Tali dispositivi vengono connessi tra loro per formare delle reti di sensori wireless (WSN) che possono essere usate per molteplici applicazioni in diversi campi quali ambientale, medico e militare.

I nodi di una WSN hanno lo scopo di monitorare l'ambiente circostante, raccogliere dati attraverso i sensori (ad esempio umidità, pressione, temperatura, ecc.) e trasmetterli a dei nodi centrali di raccolta dati (*sink*). In molti contesti applicativi tali dispositivi sono alimentati da una piccola batteria e dispongono di una capacità energetica limitata che viene consumata ad ogni operazione di misurazione, ricezione e trasmissione. Dal momento che i sensori sono spesso usati per monitorare aree remote difficili da raggiungere, ricaricare o sostituire le batterie esaurite non è sempre possibile. Per questo motivo, il risparmio energetico è uno dei requisiti primari delle reti di sensori wireless.

Uno degli obiettivi di progetto dello standard IEEE 802.15.4 [1], largamente utilizzato per le WSN, è il supporto a dispositivi con requisiti di consumo limitato della batteria. Nel 2012 è stato rilasciato il nuovo standard IEEE 802.15.4e [2], che estende la precedente versione di IEEE 802.15.4-2011 introducendo vari miglioramenti, tra cui la nuova modalità di funzionamento Time Slotted Channel Hopping (TSCH). Quando la rete opera in modalità TSCH, il tempo viene suddiviso in fasce temporali (*timeslot*) larghe abbastanza da permettere la trasmissione di un frame dati e del relativo frame di ACK. I nodi di una rete TSCH possono comunicare solo nei timeslot che gli sono stati riservati e questo permette di aumentare la prevedibilità del traffico e, soprattutto, di ridurre i consumi energetici della rete. Infatti, i dispositivi attivano l'interfaccia wireless solo durante i timeslot a loro riservati e la disattivano nel tempo rimanente per risparmiare energia.

L'introduzione della modalità operativa TSCH ha permesso di limitare considerevolmente i consumi dei nodi delle WSN. Ciononostante, in molte situazioni

una quantità non trascurabile di energia viene comunque sprecata a causa dell'*idle listening*. Questo fenomeno si verifica ogni volta che il ricevitore attiva l'interfaccia wireless in ricezione per ascoltare il canale in attesa di eventuali frame dati in arrivo, ma il trasmettitore non ha alcun frame in coda da inviare. L'idle listening rappresenta una delle principali cause del consumo energetico nelle WSN. Infatti, le reti TSCH vengono generalmente configurate per fare in modo che i timeslot riservati a un nodo si ripetano con un periodo di 1 s o 2 s. Dal momento che le trasmissioni in reti di sensori wireless avvengono tipicamente a intervalli di tempo dell'ordine di minuti (ad esempio 1 min o 10 min), un'importante numero di timeslot rimangono inutilizzati e, di conseguenza, una notevole quantità di energia viene sprecata.

Una delle tecniche proposte per la riduzione degli sprechi energetici dovuti all'idle listening è Proactive Reduction of Idle Listening (PRIL) [3]. Quest'ultima permette di estendere il funzionamento del TSCH disabilitando temporaneamente l'ascolto del canale da parte del ricevitore nei timeslot in cui non ci si aspetta di ricevere alcun frame dati dal nodo trasmettitore. La prima implementazione di PRIL proposta è PRIL-F, la quale permette di ridurre drasticamente lo spreco di energia dovuto al fenomeno di idle listening. Questa tecnica, però, è in grado di agire solo sui nodi della rete che si trovano al primo hop nel percorso di un pacchetto dal nodo sorgente al nodo destinazione. Per questo motivo, PRIL-F permette di ottenere ottimi risultati in termini di energia risparmiata solo se usata in reti in cui i percorsi sono formati da pochi hop (ad esempio reti con topologia a stella) oppure in reti con caratteristiche specifiche.

Il lavoro svolto in questa tesi ha l'obiettivo di progettare e implementare una tecnica PRIL che sia in grado di superare i limiti di PRIL-F al fine di aumentare ulteriormente l'efficienza energetica delle WSN. La tecnica proposta si chiama PRIL-MH e permette di ridurre i consumi legati al fenomeno di idle listening anche sui nodi della rete che si trovano oltre il primo hop. La soluzione ideata in PRIL-MH prevede che la disattivazione sia basata solo sull'arrivo previsto dei pacchetti caratterizzati dal periodo di generazione minimo che, solitamente, sono quelli ad avere maggiori vincoli sul tempo di consegna. Questo è soprattutto importante nelle applicazioni di tipo industriale in cui anche per la parte di rete wireless ci sono stringenti requisiti temporali [4, 5]. Ciò implica che tale tecnica possa essere usata solo quando è possibile dedurre lo schema del traffico in modo tale che sia possibile prevedere il tempo di arrivo dei pacchetti. Tuttavia, il tipo di traffico usato in molte applicazioni reali di reti TSCH è periodico e, in questi casi, il calcolo del periodo di disattivazione è immediato.

La prima versione di PRIL-MH sviluppata all'interno del lavoro di tesi è PRIL-MHB. Quest'ultima permette di ridurre in modo considerevole lo spreco di energia legato al fenomeno di idle listening, ma ha lo svantaggio di peggiorare le latenze di comunicazione. Per questo motivo, al fine di limitare i ritardi nella consegna dei pacchetti, sono state sviluppate due ulteriori versioni di PRIL-MH:

PRIL-MHI e PRIL-MHI2. In queste tecniche sono stati introdotti dei meccanismi che, rispetto a PRIL-MHB, permettono di ridurre la latenza di comunicazione media, pur mantenendo gli sprechi energetici contenuti.

L'efficacia delle tecniche è stata valutata attraverso una serie di simulazioni in cui le tre versioni di PRIL-MH sono state comparate con PRIL-F e con il TSCH Standard. A tal proposito è stato usato il simulatore ad eventi discreti `TSCH-predictor`, già sviluppato e usato in [3] per valutare le prestazioni di PRIL-F. Il simulatore è stato opportunamente modificato per poter includere anche le nuove tecniche PRIL-MH. I risultati ottenuti dalle simulazioni eseguite su reti con topologie ad albero multi-livello mostrano che le tecniche PRIL-MH sono molto efficaci nel limitare gli sprechi energetici e, infatti, permettono di ridurre drasticamente il consumo generale della rete.

La tesi è strutturata come segue. Il Capitolo 2 offre una panoramica sullo standard IEEE 802.15.4, includendo una descrizione dettagliata della modalità operativa TSCH. Le tecniche PRIL e PRIL-F sono presentate nel Capitolo 3, mentre le tre versioni della tecnica PRIL-MH (PRIL-MHB, PRIL-MHI e PRIL-MHI2) sono introdotte nel Capitolo 4. La descrizione del simulatore `TSCH-predictor` e la spiegazione delle modifiche apportate sono riportate nel Capitolo 5. I risultati ottenuti dalle simulazioni riguardanti la latenza di comunicazione e il consumo energetico sono presentati nel Capitolo 6. Infine, nel Capitolo 7 vengono tratte le conclusioni e sono forniti alcuni spunti su sviluppi futuri.

Capitolo 2

Reti IEEE 802.15.4

2.1 Applicazioni

Lo standard IEEE 802.15.4 è largamente usato per le Wireless Sensor Network (WSN), ossia reti formate da piccoli dispositivi wireless, spesso alimentati a batteria, che sono dotati di sensori in grado di misurare determinate condizioni ambientali (temperatura, pressione, umidità, livello di inquinamento, ecc.). I valori delle misurazioni vengono trasmessi ad uno o più nodi centrali, detti *sink*, che hanno il compito di raccogliere i dati ricevuti dai sensori e di trasmetterli a un server remoto. Quando i dispositivi sono anche in grado di compiere delle azioni, essi vengono definiti attuatori e si parla di Wireless Sensor and Actuator Network (WSAN).

Grazie alla loro versatilità, le WSN, e più in generale le WSAN, possono essere usate in molteplici contesti applicativi. Il più comune è il monitoraggio ambientale dove i sensori effettuano costantemente rilevazioni delle condizioni ambientali di interesse. Molto importante in questo ambito è l'utilizzo delle WSN per la rilevazione e la gestione di disastri ambientali quali incendi, frane, terremoti e inondazioni. Nelle regioni soggette a calamità naturali, i sensori possono essere usati per raccogliere misurazioni di vari parametri ambientali sulle quali è possibile sviluppare un sistema di allerta. Ad esempio, per la gestione delle inondazioni, i sensori vengono posizionati nei fiumi per monitorare il livello e la pressione dell'acqua. In caso di anormalità nei valori rilevati, vengono trasmessi dei messaggi per segnalare il pericolo in modo tale che possano essere prese delle misure precauzionali che permettano di ridurre al minimo i danni [6].

Un altro utilizzo importante è il monitoraggio del livello di qualità dell'acqua. Quest'ultima, oltre ad essere essenziale per gli esseri umani, è una sostanza fondamentale per molte attività, tra cui l'agricoltura e l'allevamento di animali. I sensori vengono usati per monitorare le proprietà dell'acqua di fiumi, laghi e falde acquifere e permettono di raccogliere dati anche in zone di difficile accesso in cui non è possibile effettuare misurazioni manualmente [7].

Oltre al monitoraggio ambientale, le WSN vengono usate anche per la domotica. Un impiego molto comune in questo ambito è il controllo dei sistemi HVAC, ossia quei sistemi che si occupano del riscaldamento, ventilazione e condizionamento dell'aria. In base al numero di persone e ai valori di temperatura e di umidità rilavati dai sensori, vengono prese delle decisioni in tempi rapidi che permettono sia di mantenere l'ambiente confortevole sia di migliorare i consumi energetici della struttura [8].

Le WSN possono anche essere usate in ambito militare per creare sistemi di sorveglianza. In questo caso vengono usati vari tipi di sensori, tra cui sensori di movimento, sensori acustici e sensori sismici. Tali sensori vengono distribuiti ai confini delle aree da proteggere con lo scopo di rilevare eventuali intrusioni nemiche [9].

Sempre più diffuse sono le Wireless Body Area Network (WBAN) per applicazioni mediche. Tali reti sono formate da sensori miniaturizzati che possono essere indossati sulla superficie del corpo oppure possono essere impiantitati direttamente nel corpo umano. Le WBAN permettono di monitorare in tempo reale lo stato di salute del paziente in mobilità e di localizzarne la posizione in caso di emergenza [10, 11].

In ambito urbano, le WSN possono essere usate per controllare il traffico e migliorare la circolazione in città. È infatti possibile sviluppare un sistema di controllo dei semafori intelligente posizionando i nodi wireless in prossimità degli incroci per rilevare la presenza di veicoli. Tali nodi inviano i dati raccolti alla stazione base, che è in grado di controllare in modo appropriato i semafori per ridurre gli ingorghi stradali causati dai tempi di attesa troppo lunghi al semaforo [12].

2.2 Standard

Lo standard IEEE 802.15.4 [1], pubblicato per la prima volta nel 2003, definisce le specifiche del livello fisico (PHY) e del sottolivello Medium Access Control (MAC) per le reti di tipo Low-Rate Wireless Personal Area Network (LR-WPAN). Tale standard è stato progettato per soddisfare i requisiti di consumo energetico limitato dei dispositivi alimentati a batteria.

2.2.1 Full Function Device e Reduced Function Device

In un rete IEEE 802.15.4 i nodi sono classificati in Full Function Device (FFD) e Reduced Function Device (RFD). Un FFD implementa tutte le funzionalità definite nello standard ed è in grado di operare come coordinatore nella WPAN (*PAN coordinator*). Gli RFD sono invece dei nodi a bassa complessità pensati per svolgere

compiti estremamente semplici, come ad esempio leggere i valori da un sensore. Gli RFD sono nodi terminali della rete e non possono ricoprire il ruolo di coordinator.

2.2.2 Topologie di rete

FFD e RFD possono essere messi in comunicazione per formare una rete secondo una delle seguenti due topologie: topologia a stella o topologia peer-to-peer. Nella topologia a stella tutti i dispositivi sono connessi a un singolo nodo centrale che opera come PAN coordinator: il primo FFD ad attivarsi stabilisce la rete e ne diventa il coordinatore. La topologia peer-to-peer, al contrario, è utilizzata in scenari più complessi e permette a un dispositivo di poter comunicare potenzialmente con qualsiasi altro nodo della stessa rete che si trova all'interno del suo raggio di comunicazione radio. Un esempio di uso della topologia peer-to-peer è il *cluster tree*, in cui qualsiasi FFD può diventare coordinator ed essere responsabile di un sottoinsieme dei nodi nella rete (*cluster*). Solo uno tra tutti i coordinator è il PAN coordinator principale, che ha il compito di stabilire la struttura iniziale della rete formando il primo cluster a cui tutti gli altri cluster possono unirsi. La struttura *multi-cluster* permette la creazione di reti complesse in grado di coprire zone molto ampie sfruttando la comunicazione multi-hop, con lo svantaggio, però, di incrementare le latenze di trasmissione.

2.2.3 Funzionalità del sottolivello MAC

A livello MAC lo standard definisce due modalità di funzionamento: *beacon-enabled* e *nonbeacon-enabled*. In modalità nonbeacon-enabled non c'è nessuna sincronizzazione tra i nodi della rete, che usano il protocollo Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) per accedere al canale di comunicazione e trasmettere i dati. Quando i dispositivi operano in modalità beacon-enabled, le trasmissioni sono regolate da una particolare struttura chiamata *superframe* (Figura 2.1), delimitata da appositi frame di segnalazione (*beacon frame*). I beacon sono spediti a intervalli regolari dal coordinator e hanno una duplice funzione: mantenere sincronizzati i nodi della rete e trasportare le informazioni relative al superframe.

Il superframe è formato da una parte attiva (*active portion*), in cui i dispositivi possono inviare o ricevere dati, e da una parte inattiva (*inactive portion*) in cui non sono permesse trasmissioni e i dispositivi hanno la possibilità di entrare in modalità risparmio energetico. La parte attiva è a sua volta suddivisa equamente in 16 *timeslot* elementari e inizia sempre con la trasmissione del beacon frame a cui segue un Contention Access Period (CAP). Durante questo periodo l'accesso al canale per la trasmissione delle informazioni è regolamentato dal protocollo CSMA/CA.

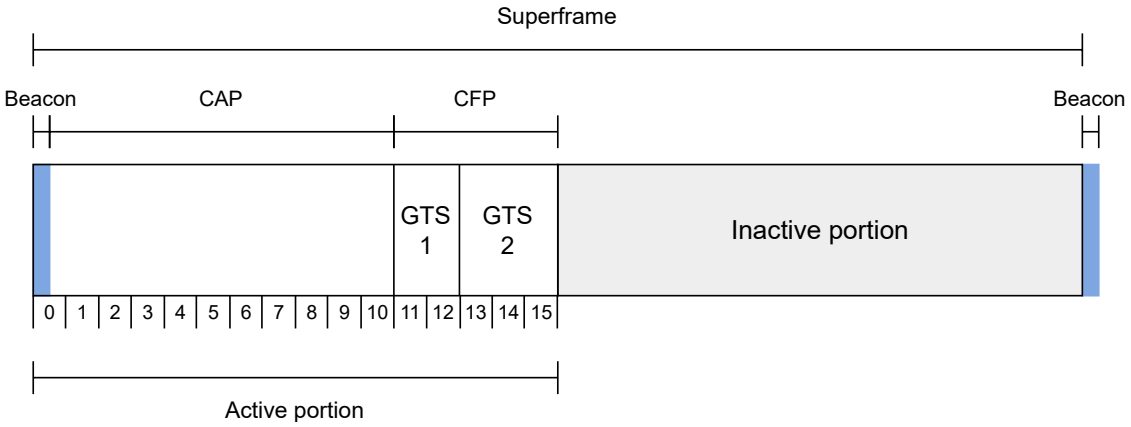


Figura 2.1: Esempio di struttura superframe

Per supportare applicazioni in cui la velocità, l'affidabilità e il determinismo sono requisiti fondamentali, il CAP può essere opzionalmente seguito da un Contention Free Period (CFP). In questo caso un dispositivo può richiedere al PAN coordinator che gli venga riservata una porzione dell'active superframe. Tale porzione prende il nome di Guaranteed Time Slot (GTS) e può essere composta da uno o più timeslot. Se ci sono abbastanza timeslot liberi per soddisfare la richiesta, il coordinator procede con l'allocazione del GTS e genera il *GTS descriptor*. Il GTS descriptor contiene le informazioni sul GTS, tra cui il timeslot di inizio e la lunghezza, e viene incluso nel beacon frame. Quando un GTS non è più necessario si procede con la sua deallocazione rimuovendo il relativo GTS descriptor dal beacon frame. Un GTS può essere deallocato in qualsiasi momento a discrezione del coordinator oppure a seguito di una richiesta da parte del dispositivo per cui era stato allocato. L'insieme di tutti i GTS allocati forma il CFP. In modalità beacon-enabled non sono permesse comunicazioni che non includano il coordinator e, pertanto, le topologie possibili si limitano a quelle di tipo gerarchico (ad esempio topologia a stella e topologia ad albero).

2.2.4 Standard 802.15.4e e DSME

Nel 2012 IEEE ha rilasciato lo standard IEEE 802.15.4e [2] come estensione del precedente standard IEEE 802.15.4-2011 con l'obiettivo di fornire un supporto migliore alle applicazioni del settore industriale. Lo standard IEEE 802.15.4e introduce miglioramenti nel sottolivello MAC, definendo nuove modalità operative, tra cui Deterministic and Synchronous Multi-channel Extension (DSME) e Time Slotted Channel Hopping (TSCH). DSME e TSCH sono state integrate nello standard a partire dalla successiva rivisitazione IEEE 802.15.4-2015 e sono mantenute nell'attuale versione IEEE 802.15.4-2020.

DSME estende la modalità beacon-enabled già presente nello standard 802.15.4-2011, usando una nuova struttura temporale, chiamata *multi-superframe* (Figura 2.2), composta da una sequenza di superframe. In modo simile alla struttura definita nella versione IEEE 802.15.4-2011, ogni superframe in modalità DSME è diviso in 16 timeslot di egual durata ed è composto da un beacon frame, un CAP e un CFP. Il CAP inizia sempre dopo la trasmissione del beacon e termina prima del timeslot n° 9, ed è seguito dal CFP che, invece, è composto da 7 timeslot.

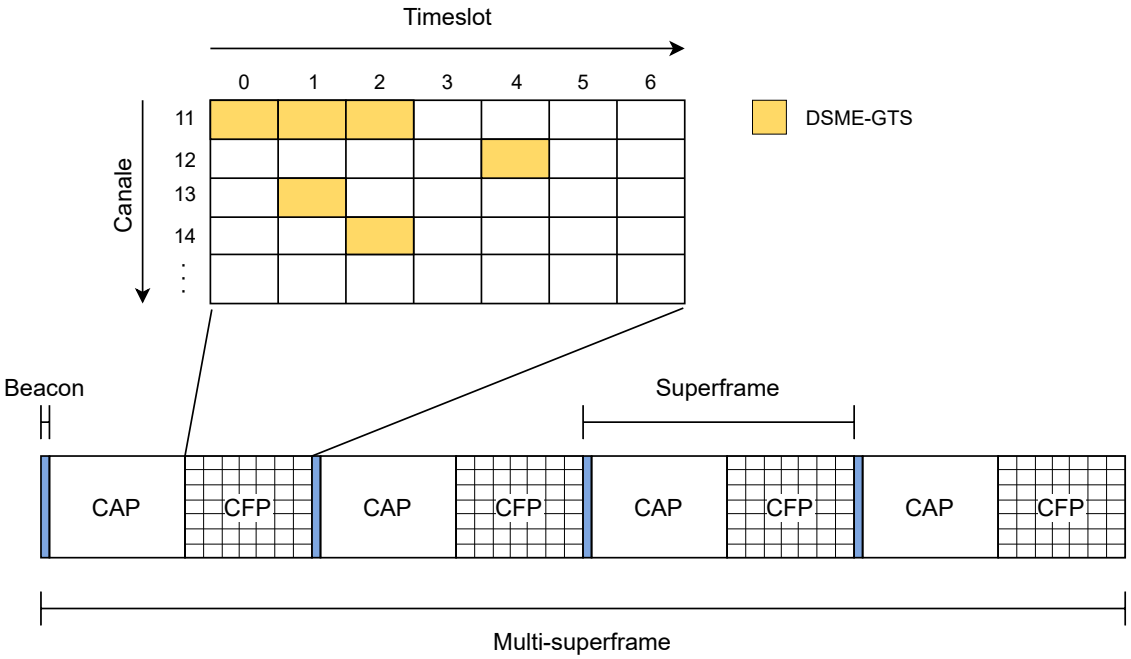


Figura 2.2: Esempio di struttura multi-superframe DSME

A differenza di quanto accade nella modalità beacon-enabled, durante il CFP definito dal DSME, più dispositivi possono trasmettere contemporaneamente sfruttando frequenze diverse. Questo è possibile grazie ai *DSME-GTS*, che estendono le funzionalità già offerte dai GTS tradizionali, rendendole compatibili con la comunicazione multi-canale. Per ogni DSME-GTS, infatti, viene specificato il canale da usare per la comunicazione, permettendo così a più dispositivi di allocare un DSME-GTS sullo stesso periodo temporale, ma su canali differenti in modo da garantire trasmissioni senza collisioni. Dunque, come illustrato in Figura 2.2, il CFP può essere visto come una griglia in cui le colonne rappresentano i timeslot, le righe rappresentano i canali di comunicazione e le celle libere rappresentano le coppie <timeslot, canale> in cui possono essere allocati nuovi DSME-GTS.

Un ulteriore vantaggio offerto dal DSME è quello di poter allocare DSME-GTS a una coppia di dispositivi all'interno del raggio di comunicazione radio per permettergli di comunicare direttamente, eliminando così la necessità di scambiare messaggi

attraverso il coordinator. Inoltre, il processo di allocazione di un DSME-GTS, così come quello di deallocazione, non richiede l'intervento del coordinator ed è completamente gestito dagli stessi dispositivi che lo richiedono: il nodo sorgente invia una richiesta al nodo destinazione, il quale, dopo i dovuti controlli, procede con l'allocazione del DSME-GTS e notifica la sorgente. Pertanto, a differenza della modalità beacon-enabled, le topologie di rete possibili non si limitano solo a quelle di tipo gerarchico, ma includono anche le configurazioni a maglia parzialmente connessa e completamente connessa.

Inoltre, per aumentare l'affidabilità e ridurre i problemi dovuti alle interferenze, il protocollo DSME fornisce due meccanismi di diversificazione del canale (channel diversity): *channel adaptation* e *channel hopping*.

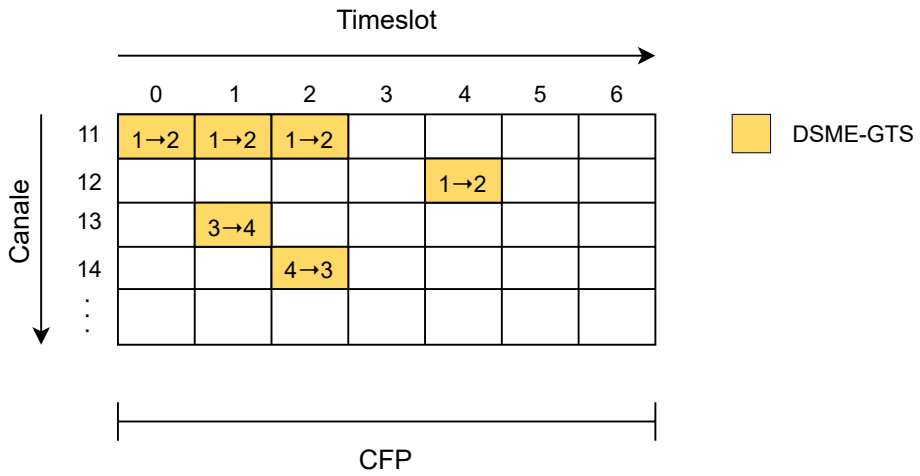


Figura 2.3: Esempio di allocazione dei DSME-GTS con channel adaptation

Nel channel adaptation, i DSME-GTS per una coppia di dispositivi possono essere allocati su un singolo canale o su canali differenti, scelti in base alla loro qualità attuale. La Figura 2.3 ne illustra un esempio, in cui i DSME-GTS per i nodi 1 e 2 sono stati allocati usando due canali differenti: il canale 11 dal timeslot 0 al timeslot 2 e il canale 12 nel timeslot 4. Il canale assegnato a un DSME-GTS non cambia nel tempo, a patto che la qualità del segnale ricevuto non peggiori. Infatti, quando la qualità del collegamento di un DSME-GTS scende al di sotto di una certa soglia, questo viene deallocato per allocarne uno nuovo su un altro canale in cui ci si aspetta una qualità del segnale migliore.

Nel channel hopping, a ogni nodo viene indicato il canale in cui gli è permesso allocare un DSME-GTS (in ricezione). Il canale che il nodo può usare cambia ad ogni timeslot seguendo un ordine predefinito. La serie ordinata di canali prende il nome di *Hopping Sequence* e si ripete ciclicamente nel tempo. A ogni dispositivo viene inoltre assegnato un *channel offset* diverso, che indica la posizione di partenza nella Hopping Sequence. La Figura 2.4 mostra un esempio della tecnica channel

hopping per due dispositivi e, per ogni timeslot, è evidenziato il canale da usare. Nell'esempio in questione la Hopping Sequence è {11, 12, 13, 14, 15} e i channel offset assegnati al dispositivo 1 e al dispositivo 2 sono rispettivamente 2 e 0.

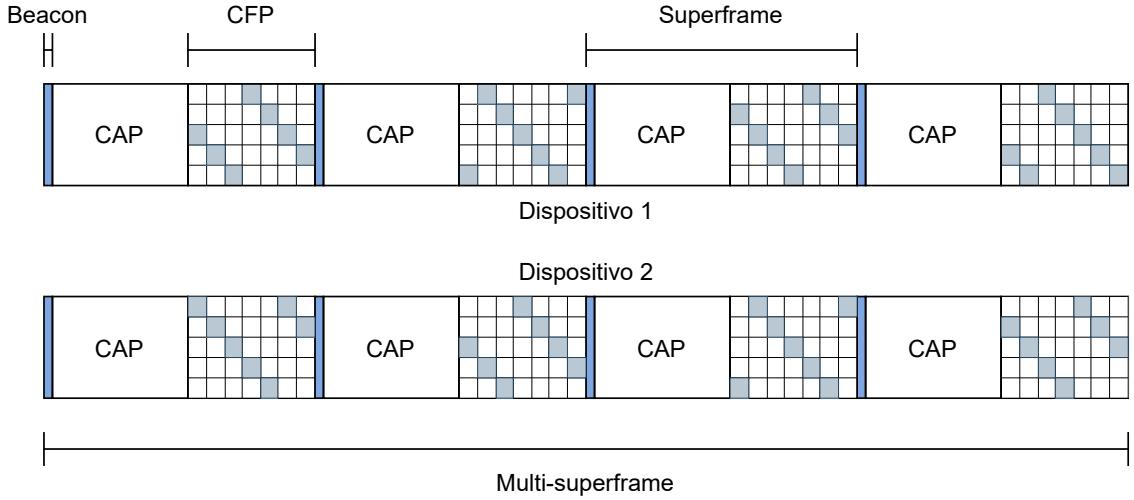


Figura 2.4: Esempio di allocazione dei DSME-GTS con channel hopping

In aggiunta al channel hopping, DSME prevede altre innovazioni che permettono di migliorare specifici indicatori prestazionali. Per esempio, in contesti in cui l'efficienza energetica è un requisito primario, DSME definisce un meccanismo chiamato *CAP reduction* (Figura 2.5). Quando abilitato, il CAP rimane presente solo nel primo superframe del multi-superframe e viene rimosso dai superframe rimanenti in favore di un CFP più lungo.

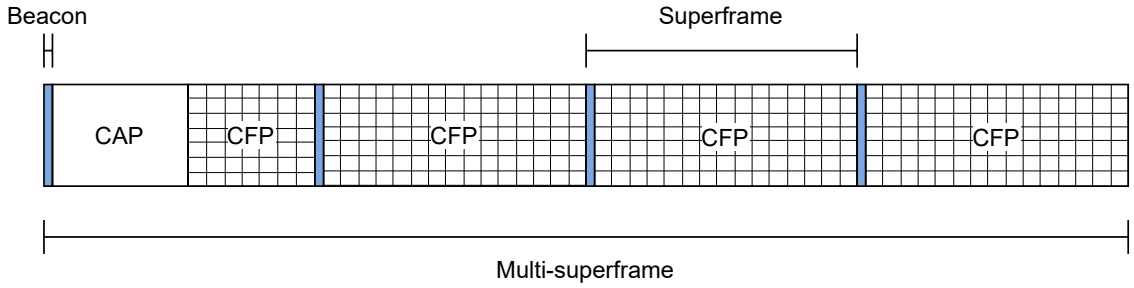


Figura 2.5: Struttura multi-superframe DSME quando il CAP reduction è attivo

L'altra modalità operativa definita a partire dallo standard IEEE 802.15.4e, il TSCH, combina il meccanismo di accesso *time slotted* con quello di *channel hopping* per offrire alta affidabilità, latenze prevedibili ed efficienza energetica. In modalità TSCH, il superframe è sostituito dallo *slotframe*, ossia una collezione di timeslot che si ripete nel tempo. Per evitare collisioni e ridurre il numero di ritrasmissioni, possono essere riservati dei timeslot a coppie di dispositivi. TSCH

non impone nessun limite sulla topologia e può essere usato per formare qualsiasi tipo di rete, da quelle più semplici con topologia a stella fino a quelle più complesse con topologia a maglia completamente connessa. La modalità operativa TSCH verrà dettagliatamente discussa in Sezione 2.3.

2.2.5 Livello PHY

Lo standard IEEE 802.15.4 definisce diverse possibili implementazioni del livello PHY, ognuna delle quali differisce dalle altre per le bande di frequenza su cui opera e per il tipo di modulazione usato.

L'implementazione più diffusa è O-QPSK PHY, che è basata sullo schema di modulazione Offset Quadrature Phase-Shift Keying (O-QPSK) e può operare su tre bande di frequenza differenti: 868 MHz, 915 MHz e 2.4 GHz. Quella più nota e maggiormente utilizzata è la banda ISM 2.4 GHz, che offre un data rate di 250 Kbps ed è formata da 16 canali di comunicazione disposti nell'intervallo tra 2405 MHz (channel 11) e 2480 MHz (channel 26), ognuno dei quali con una larghezza di banda di 5 MHz.

La tecnica di modulazione adottata da O-QPSK PHY è basata su 16 sequenze quasi ortogonali di tipo Pseudo-random Noise (PN). In trasmissione, l'informazione binaria del frame viene processata in gruppi di 4 bit che prendono il nome di *data symbol*. Ogni data symbol viene utilizzato per selezionare una delle 16 sequenze PN lunghe 32 chip, come riportato in Tabella 2.1. Tutte le sequenze associate a data symbol consecutivi sono concatenate e formano una sequenza di chip, che viene modulata sull'onda portante attraverso la tecnica O-QPSK. In ricezione il segnale viene demodulato per riottenere il flusso di chip trasmesso. Ogni sequenza ricevuta è confrontata con ognuna delle 16 valide, selezionando come sequenza PN trasmessa quella con la distanza di hamming minore. Infine, la sequenza selezionata viene convertita nel data symbol corrispondente per ricostruire l'informazione binaria del frame.

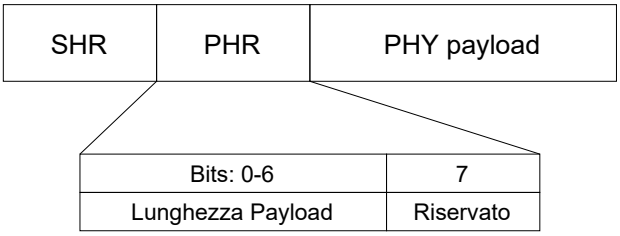


Figura 2.6: Struttura del PHY Protocol Data Unit

La struttura del frame è stata progettata per essere semplice e allo stesso tempo sufficientemente robusta per supportare comunicazioni su canali rumorosi. Il frame del livello fisico (Figura 2.6), chiamato PHY Protocol Data Unit (PPDU), è

formato da tre sezioni: Synchronization Header (SHR), PHY Header (PHR), PHY Payload. Solo 7 bit del PHR possono essere usati per specificare la lunghezza del PHY payload e pertanto la dimensione massima di dati che il PPDU è in grado di trasportare è di 127 byte.

Data symbol	**Sequenza di chip c0 cl ... c30 c31**
0	11011001110000110101001000101110
1	11101101100111000011010100100010
2	00101110110110011100001101010010
3	00100010111011011001110000110101
4	01010010001011101101100111000011
5	00110101001000101110110110011100
6	11000011010100100010111011011001
7	10011100001101010010001011101101
8	10001100100101100000011101111011
9	10111000110010010110000001110111
10	01111011100011001001011000000111
11	01110111101110001100100101100000
12	00000111011110111000110010010110
13	01100000011101111011100011001001
14	10010110000001110111101110001100
15	11001001011000000111011110111000

Tabella 2.1: Mapping tra data symbol e sequenza PN

Oltre alla trasmissione e alla ricezione dei frame, il livello PHY è responsabile di diversi compiti quali la selezione della frequenza del canale, la misurazione del Link Quality Indicator (LQI) e l'esecuzione del Clear Channel Assessment (CCA).

LQI è utilizzato per caratterizzare la qualità di un pacchetto ricevuto. La misurazione può essere implementata attraverso il processo di Energy Detection (ED), che misura la potenza del segnale ricevuto all'interno della banda del canale, oppure usando una stima del rapporto segnale/rumore (SNR). Indipendentemente dal metodo di misurazione scelto (ED o SNR), il risultato ottenuto viene mappato sull'intervallo 0-255, in cui il valore minimo (0) e il valore massimo (255) coincidono

rispettivamente con la peggiore e la migliore qualità del segnale che il destinatario è in grado di rilevare.

CCA è un meccanismo usato dal protocollo CSMA/CA per assicurarsi che il canale di comunicazione sia libero prima di una trasmissione. Per essere conformi allo standard IEEE 802.15.4, i dispositivi devono implementare almeno una delle seguenti modalità CCA:

- **Energy above threshold**: in questa modalità viene preso in considerazione il risultato ottenuto dal processo di ED. Se il valore è superiore a una certa soglia (*ED threshold*) allora il canale è considerato occupato.

- **Carrier sense only**: in questa modalità vengono analizzare le caratteristiche del segnale occupante, tra cui il tipo di modulazione usata. Il canale è considerato occupato solo se il segnale rilevato è conforme allo standard.

- **Carrier sense with energy above threshold**: per determinare lo stato di occupazione del canale viene usata una combinazione logica (AND o OR) delle modalità Energy above threshold e Carrier sense only.

2.3 TSCH

La modalità operativa TSCH è stata definita per la prima volta nello standard IEEE 802.15.4e rilasciato nel 2012. La tecnica di accesso *time slotted* e il *channel hopping* sono le caratteristiche principali del TSCH. La prima permette di aumentare la prevedibilità del traffico e di ridurre il consumo di energia, mentre la seconda è un metodo efficace per mitigare gli effetti negativi delle interferenze causate dai dispositivi esterni alla rete che operano sulle stesse frequenze.

2.3.1 Accesso Time Slotted

Nel TSCH, il tempo è diviso in timeslot della stessa durata, i quali sono raggruppati in uno *slotframe* (Figura 2.7) che ne contiene un numero predefinito. Ogni timeslot è lungo abbastanza per ospitare la trasmissione di un frame dati e del suo eventuale frame di acknowledgment (ACK). Lo standard non impone nessuna lunghezza per lo slotframe che può, quindi, variare in base ai requisiti dell'applicazione: più breve è lo slotframe, più spesso si ripeteranno i timeslot, ottenendo delle latenze più basse, ma anche un consumo energetico maggiore [13]. I timeslot in una rete TSCH sono univocamente identificati a mezzo di un contatore chiamato Absolute Slot Number (ASN), il quale è inizializzato a 0 durante la creazione della rete e incrementato di 1 ad ogni timeslot.

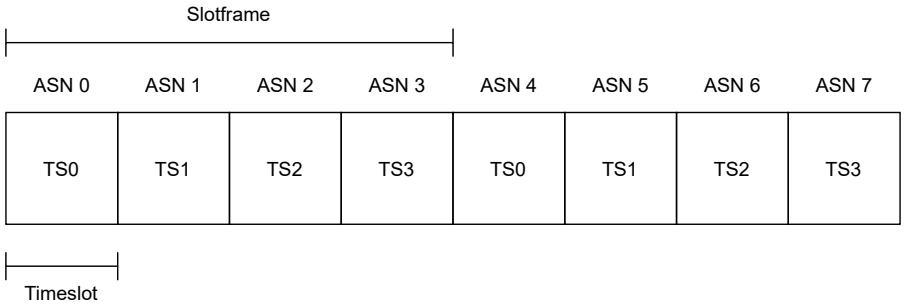


Figura 2.7: Esempio di slotframe TSCH formato da 4 timeslot

Le comunicazioni in una rete TSCH sono coordinate dallo *schedule*, che stabilisce l'utilizzo dello slotframe da parte dei nodi. Lo schedule può essere visto come una matrice in cui le colonne rappresentano i timeslot nello slotframe e le righe rappresentano i canali di comunicazione disponibili. Una cella all'interno dello schedule è identificata da uno *slotOffset* e da un *channelOffset* che indicano rispettivamente la colonna e la riga nella matrice [14, 15, 16].

Le celle possono essere configurate in modo da ottenere lo schedule desiderato e, in particolare, possono essere allocate a coppie di nodi per garantirgli accesso esclusivo al mezzo di comunicazione. In questo caso, le celle vengono definite *scheduled* e i dispositivi comunicano usando il timeslot e il canale a loro riservati. Un esempio di schedule è mostrato in Figura 2.8.

Dal punto di vista di un singolo dispositivo, una cella scheduled può essere configurata o in trasmissione (TX) o in ricezione (RX). In quelle configurate in trasmissione, il dispositivo trasmette il primo frame dati in coda per il nodo destinazione indicato nello schedule e attende la ricezione di un ACK. Nel caso in cui non fosse presente alcun frame da trasmettere, il dispositivo può disattivare il ricetrasmettitore per tutta la durata del timeslot. Nelle celle in ricezione, il dispositivo si mette in ascolto per possibili frame in arrivo. Se viene ricevuto un frame dati, il nodo procede con la trasmissione di un ACK, altrimenti, dopo un periodo iniziale di attesa, può spegnere il ricevitore radio fino alla fine del timeslot. Nei timeslot in cui non è riservata alcuna cella, il dispositivo disattiva il ricetrasmettitore in modo da preservare la durata della batteria. Il TSCH prevede anche la possibilità di avere delle celle *shared* in cui più dispositivi possono trasmettere. Per evitare possibili collisioni, è stato definito un algoritmo di backoff per le trasmissioni in celle shared.

In caso di traffico elevato, più celle dello slotframe, dette *bundle*, possono essere riservate per le comunicazioni tra due nodi della rete. Questa tecnica è chiamata *overprovisioning* e ha il vantaggio di mantenere le latenze basse, anche nelle situazioni in cui una scarsa qualità del canale di comunicazione provoca frequenti ritrasmissioni. In questo caso, i dispositivi per i quali è stato allocato un bundle hanno la possibilità di effettuare più ritrasmissioni all'interno dello stesso slotframe

e, quindi, i frame dati possono arrivare a destinazione in tempi più brevi. Sfortunatamente, l'overprovisioning ha lo svantaggio di incrementare il consumo energetico del ricevitore, a patto che non si realizzino specifiche tecniche che gestiscano lo spegnimento delle celle sicuramente non utilizzate [17]. Infatti, anche se non ci sono frame da trasmettere, il ricevitore rimane comunque attivo in tutte le celle del bundle.

L'algoritmo di scheduling che si occupa di riservare e configurare le celle della matrice non è parte dello standard e la sua implementazione è lasciata ai livelli superiori dello stack protocollare.

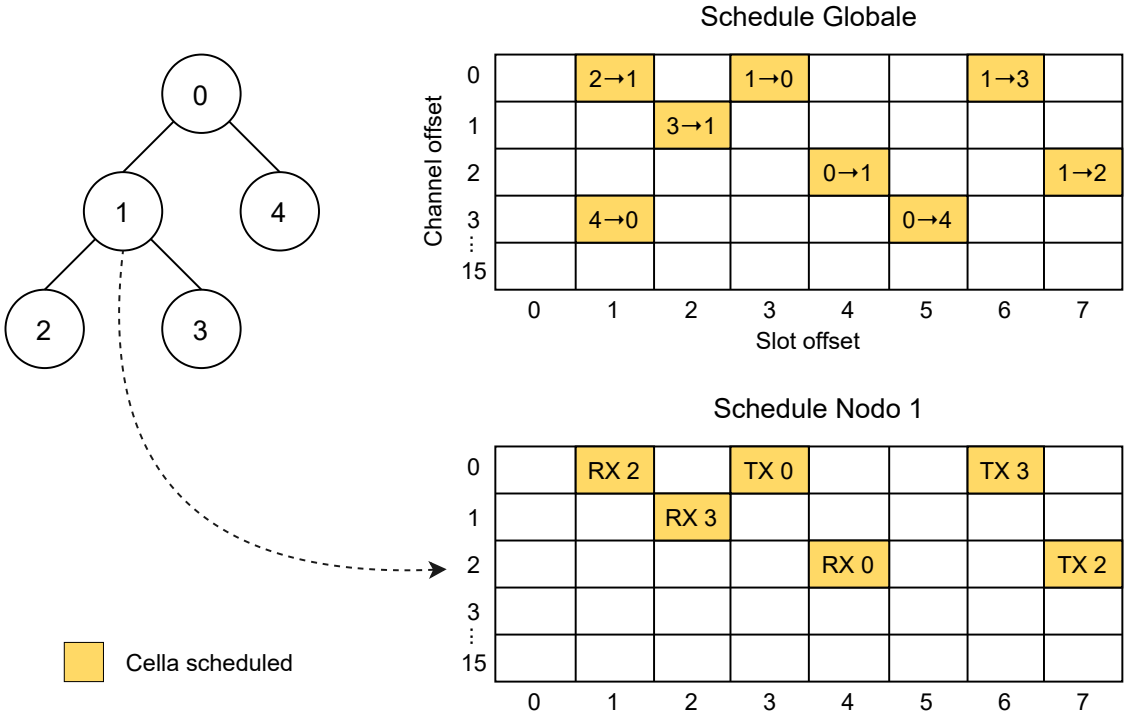


Figura 2.8: Esempio di schedule di una rete TSCH composta da cinque nodi. Oltre allo schedule globale (matrice in alto), è riportato anche lo schedule locale del Nodo 1 (matrice in basso) in cui le celle allocate sono etichettate con TX (Nodo 1 trasmettitore) o RX (Nodo 1 ricevitore) seguito dall'identificativo dell'altro nodo coinvolto nella comunicazione

2.3.2 Channel Hopping

Oltre alla tecnica di accesso time slotted, il TSCH fornisce un meccanismo di diversificazione di canale, il channel hopping, che permette di raggiungere un elevato livello di affidabilità riducendo l'impatto delle interferenze esterne. Infatti, se una trasmissione tra due dispositivi fallisce, ritrasmettere su un canale diverso aumenta

la probabilità di successo rispetto a riutilizzare la stessa frequenza. Per questo motivo, la frequenza usata dai dispositivi per le trasmissioni in una data cella cambia ad ogni ripetizione dello slotframe ed è selezionata in base all'ASN e al channelOffset associato alla cella come segue:

$$f = HoppingSequence[\,(ASN + channelOffset)\ \%\ N_{Freq}\,] \tag{2.1}$$

dove $HoppingSequence$ è una lista ordinata di canali e N_{Freq} indica il numero di canali nella lista (ad esempio 16 quando si opera nella banda 2.4 GHz e tutti i canali vengono usati). Inoltre, sfruttando i diversi valori di channelOffset, è possibile allocare più celle in uno stesso timeslot per consentire a più dispositivi di trasmettere contemporaneamente usando frequenze diverse. La Figura 2.9 illustra un esempio del meccanismo di channel hopping in cui sono riportati i canali assegnati alle celle della matrice TSCH per due slotframe consecutivi. La Hopping Sequence utilizzata è {11, 15, 26, 25, 20, 13} e, quindi, il numero di frequenze utilizzabili è $N_{Freq} = 6$. Osservando i timeslot con ASN = 1 e ASN = 9, è possibile notare come l'utilizzo di valori diversi di channelOffset permetta di avere due trasmissioni in parallelo. In particolare, dato il numero di canali disponibili ($N_{Freq} = 6$), è possibile creare fino a sei trasmissioni in parallelo in un singolo timeslot.

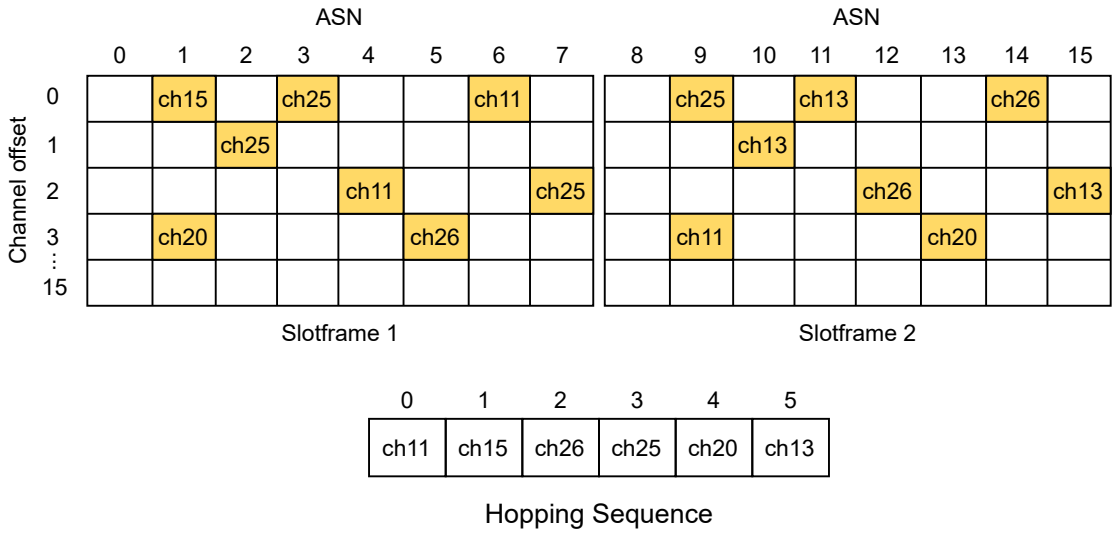


Figura 2.9: Esempio di channel hopping TSCH

2.3.3 Sincronizzazione

In una rete TSCH è necessario che i nodi rimangano sempre sincronizzati in modo tale che tutti abbiano la stessa nozione di quando uno slotframe inizia e finisce.

Per questo motivo, un dispositivo deve sincronizzare periodicamente il suo orologio interno e per farlo usa uno o più nodi vicini come riferimenti temporali.

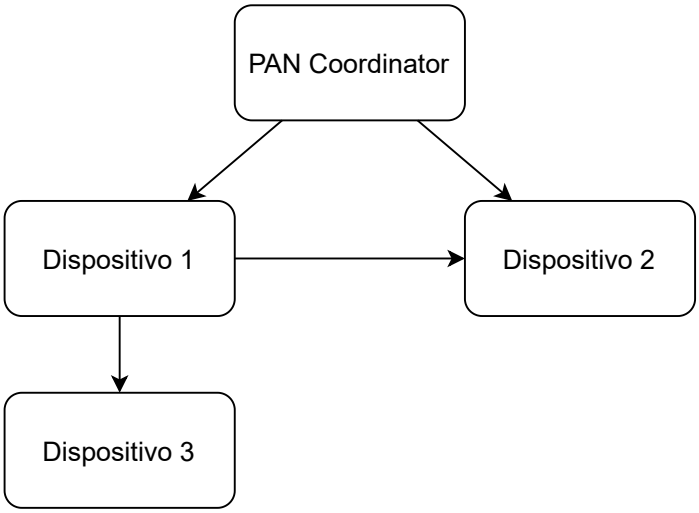


Figura 2.10: Esempio di propagazione del tempo in una rete TSCH

Lo schema in Figura 2.10 illustra i riferimenti temporali dei dispositivi di una rete TSCH. Le frecce indicano la direzione verso cui viene propagata l'informazione sul tempo. In questo esempio, il PAN coordinator agisce come sorgente temporale per l'intera rete. Il Dispositivo 1 sincronizza il suo orologio solo con quello del PAN coordinator, mentre il Dispositivo 3 usa come riferimento temporale il Dispositivo 1. Al contrario, la sincronizzazione del Dispositivo 2 dipende sia dal PAN coordinator sia dal Dispositivo 1.

Lo standard IEEE 802.15.4e definisce due tecniche di sincronizzazione, una basata sul frame di ACK e l'altra basata sul frame dati. Nel primo caso, la sincronizzazione avviene attraverso lo scambio di un frame dati e del suo relativo ACK. Il ricevitore calcola la differenza tra il tempo in cui si aspettava di ricevere il frame dati e la sua ricezione effettiva, e usa il frame di ACK per informare il trasmettitore della correzione. Se il ricevitore è un riferimento temporale per il trasmettitore, quest'ultimo regola il suo orologio includendo la differenza ricevuta in una media con le ultime correzioni ricevute dagli altri nodi riferimento.

Invece, nel metodo basato su frame dati è il ricevitore a sincronizzarsi al trasmettitore. Ogni volta che riceve un frame dati da uno dei suoi riferimenti temporali, il ricevitore calcola la differenza tra il tempo di arrivo previsto e quello effettivo, e usa questa informazione per regolare il suo orologio interno facendo una media delle differenze temporali ricevute dai suoi riferimenti.

Indipendentemente dalla tecnica adottata, il traffico dati è usato per mantenere implicitamente la sincronizzazione tra un nodo e il suo riferimento temporale. Nel caso di assenza di comunicazioni tra i due per un periodo di tempo più lungo

del *keep-alive period*, il dispositivo deve mandare un frame dati vuoto (*keep-alive message*) al nodo riferimento per innescare il processo di sincronizzazione.

Lo scambio periodico di messaggi è necessario in quanto la frequenza di oscillazione dei quarzi su cui si basa il tempo del dispositivo dipende da molti fattori quali la temperatura, le vibrazioni e la tensione di alimentazione [18, 19].

Capitolo 3

Tecniche PRIL

3.1 Introduzione

Le WSN sono molto versatili al punto da permetterne l'utilizzo in vari contesti applicativi. Questo è possibile grazie a due delle caratteristiche principali dei dispositivi usati come sensori, ossia il loro essere autonomi dal punto di vista energetico ed essere dotati di un modulo wireless per le comunicazioni. Infatti, tali dispositivi sono alimentati da una batteria e possono essere posizionati liberamente in qualsiasi luogo, senza necessità di cablaggi. Tuttavia, la riserva energetica di cui dispongono è limitata e viene consumata ad ogni operazione di misurazione, ricezione e trasmissione. Ricaricare o sostituire le batterie esaurite non è sempre possibile. Ad esempio, quando le WSN sono usate per monitorare i disastri ambientali, i sensori vengono posizionati in aree geografiche remote che non rendono agevole sostituire la batteria quando quest'ultima si scarica completamente. Diventa, quindi, necessario progettare un sistema in grado di migliorare l'efficienza energetica delle reti di sensori per aumentare il tempo di vita delle batterie.

Un primo passo verso il risparmio energetico nelle WSN è stato fatto con l'introduzione della modalità operativa TSCH dello standard IEEE 802.15.4. Essa è stata progettata proprio con lo scopo di soddisfare i requisiti di energia limitata dei dispositivi wireless usati nelle WSN. Grazie alla modalità di accesso time slotted e alla programmazione del traffico, la tecnica TSCH permette di limitare i consumi energetici dei nodi della rete. In primo luogo vengono eliminate le ritrasmissioni dovute alle collisioni. Infatti, ad ogni coppia di nodi che intende comunicare viene riservata una porzione della banda (timeslot e canale) in cui nessun altro dispositivo può trasmettere. Inoltre, un nodo di una rete TSCH non ha la necessità di mantenere l'interfaccia wireless costantemente attiva, ma può decidere di accenderla solo nei timeslot di interesse, ovvero nei timeslot in cui gli sono state allocate delle celle in trasmissione o in ricezione. Nei timeslot rimanenti dello slotframe, il dispositivo non può né trasmettere né ricevere frame dati e, pertanto, può spegnere il modulo radio, e in alcuni contesti la CPU e altre periferiche, per risparmiare energia.

Ulteriori miglioramenti in termini di efficienza energetica possono essere ottenuti in implementazioni ottimizzate del TSCH. Ad esempio, è possibile evitare di attivare l'interfaccia wireless nelle celle configurate in trasmissione nel caso in cui non ci siano frame dati in coda pronti per essere trasmessi. Invece, nelle celle riservate alla ricezione di frame dati, il nodo può decidere di spegnere il ricetrasmettitore se, dopo un periodo iniziale di ascolto del canale, non viene rilevato alcun segnale.

Nonostante la modalità operativa TSCH permetta un utilizzo efficiente della batteria dei dispositivi di una WSN, in molte situazioni una considerevole quantità di energia viene comunque sprecata. Ad esempio, in ogni timeslot in cui è allocata una cella in ricezione, il dispositivo deve sempre accendere il ricevitore radio e ascoltare il canale per eventuali frame dati in arrivo. Nei casi in cui il nodo trasmettitore non abbia alcun frame in coda da inviare al nodo ricevitore, allora l'energia usata da quest'ultimo per riattivare il modulo radio e ascoltare il canale è da considerarsi sprecata. Questo fenomeno prende il nome di *idle listening* ed è una delle principali cause di consumo energetico nei dispositivi wireless.

Analizzando i consumi dei dispositivi che operano in modalità TSCH, si ha che l'energia consumata da un nodo in idle listening, ossia per ascoltare il canale per la ricezione di un singolo frame dati in assenza di trasmissioni, può arrivare ad essere più della metà dell'energia consumata dal nodo per trasmettere o ricevere un frame dati (maggiori dettagli in Tabella 6.2 in Sezione 6).

Numerosi studi sono stati condotti con l'obiettivo di estendere il funzionamento della modalità TSCH al fine di aumentare l'efficienza energetica della rete, agendo proprio sull'idle listening. Ad esempio, in [17] sono proposte due tecniche che permettono di ridurre lo spreco di energia causato da questo fenomeno quando viene sfruttato l'overprovisioning. Quest'ultimo permette di avere all'interno di uno stesso slotframe più celle riservate per le comunicazioni tra una coppia di dispositivi. L'overprovisioning, però, porta a un incremento del consumo energetico a causa di una maggiore frequenza del fenomeno di idle listening. Infatti, anche quando non è necessario, il ricevitore deve riattivarsi in ogni cella del bundle per ascoltare il canale. Le tecniche proposte permettono di ridurre l'idle listening, disabilitando temporaneamente l'ascolto del canale da parte del ricevitore appena il trasmettitore non ha più frame dati in coda da trasmettere. Il ricevitore viene informato della presenza di ulteriori frame in coda attraverso un campo aggiuntivo del frame dati, in cui il trasmettitore riporta l'informazione sullo stato della sua coda. Se in una cella non viene ricevuto alcun frame, oppure viene ricevuto un frame in cui è indicata esplicitamente l'assenza di ulteriori frame dati in coda, il ricevitore disabilita l'ascolto del canale nelle celle rimanenti del bundle. La sospensione ha effetto fino alla fine dello slotframe e il normale comportamento del ricevitore è ripristinato all'inizio dello slotframe successivo. Questo vuol dire che il ricevitore deve sempre ascoltare il canale almeno nella prima cella del bundle prima di poter disabilitare l'ascolto e, quindi, non si possono ottenere miglioramenti sui dispositivi per cui l'overprovisioning non è abilitato. L'overprovisioning ha come obiettivo

principale la riduzione delle latenze a discapito di un leggero aumento del consumo energetico.

Invece, una soluzione maggiormente vocata alla riduzione del consumo energetico è descritta in [3] e alcuni suoi miglioramenti sono descritti in [20]. La tecnica si chiama Proactive Reduction of Idle Listening (PRIL) e permette di allungare la durata delle batterie dei dispositivi riducendo il fenomeno di idle listening. Anche in questo caso l'obiettivo viene raggiunto impedendo temporaneamente al ricevitore di ascoltare il canale, ma, a differenza delle tecniche citate precedentemente, la sospensione può durare per più slotframe e questo ne permette l'utilizzo anche quando l'overprovisioning non è abilitato.

Una prima implementazione di PRIL, chiamata PRIL-F [3], è già stata sviluppata e offre un meccanismo per eliminare l'idle listening sui nodi della rete al primo hop. In questo lavoro di tesi, a partire da PRIL-F, è stata creata una tecnica che permette di usare PRIL anche nei nodi oltre il primo hop, andando a superare la più grossa limitazione di PRIL-F. Le tecnica si chiama PRIL-MH e ne sono state sviluppate tre versioni: PRIL-MHB, PRIL-MHI e PRIL-MHI2.

3.2 PRIL

PRIL è una tecnica sviluppata per limitare gli sprechi di energia causati dal fenomeno di idle listening. L'idea alla base della tecnica è quella di potenziare la modalità operativa TSCH per permettere ai dispositivi di disabilitare l'attivazione dell'interfaccia wireless nei timeslot in cui è stata allocata una cella in ricezione, ma non ci si aspetta di ricevere alcun frame dati dal nodo trasmettitore. Una caratteristica importante delle tecniche PRIL è quella che il loro utilizzo non può peggiorare le prestazioni della rete in termini di affidabilità e di consumo energetico.

Analizzando i consumi dei nodi in diversi contesti, ci si rende conto che il contributo dell'idle listening sul consumo totale di un dispositivo non è sempre uguale, ma varia in base alla frequenza con la quale vengono generati nuovi pacchetti da parte dell'applicazione. In particolare, maggiore è la frequenza con cui l'applicazione genera pacchetti, minore sarà il numero di celle che il ricevitore passerà in idle listening. Infatti, nei casi in cui il traffico è elevato, la probabilità che ci sia almeno un pacchetto in coda pronto per essere trasmesso è alta e, pertanto, la percentuale di celle non sfruttate si riduce. A dimostrazione di ciò, si considerino due semplici esempi di comunicazioni tra due nodi di una rete TSCH per i quali è stata allocata una cella che si ripete ogni 2 s. Nel primo caso l'applicazione genera pacchetti ogni $T_{app} = 60\,\text{s}$, che vuol dire che si ha un nuovo frame in coda ogni 30 ripetizioni della cella. Quindi, se la trasmissione del pacchetto ha successo al primo tentativo, solo una cella è effettivamente sfruttata per comunicare, mentre le restanti 29 rimangono inutilizzate e si ha il fenomeno di idle listening. Nel secondo caso, impostando un periodo di generazione più breve, $T_{app} = 10\,\text{s}$, si ha un nuovo

pacchetto ogni 5 ripetizioni della cella. Pertanto, nell'eventualità in cui non ci sia bisogno di ritrasmissioni, il fenomeno di idle listening si verifica solo in 4 celle, con uno spreco di energia 7.25 volte minore rispetto al primo caso.

Tuttavia, in contesti reali, i periodi con cui i dispositivi di una WSN generano nuovi pacchetti non sono tipicamente piccoli. Infatti, il numero di trasmissioni deve essere mantenuto basso per poter preservare la durata della batteria. Spesso i pacchetti vengono generati periodicamente, con periodi che variano da pochi minuti fino ad arrivare a diverse ore. In questi casi il traffico è decisamente limitato e, di conseguenza, lo spreco di energia causato dall'idle listening è considerevole.

La tecnica PRIL è stata progettata per essere efficace contro l'idle listening e allo stesso tempo facilmente implementabile sui dispositivi usati nelle WSN, che sono caratterizzati da risorse computazionali limitate. Questa tecnica opera a livello MAC, nel quale viene definito un nuovo comando, chiamato comando di *sleep*. Quest'ultimo può essere incluso direttamente nei pacchetti scambiati normalmente tra due nodi. In questo modo si ha un doppio vantaggio, ossia quello di non aumentare in modo consistente la banda e, soprattutto, quello di poter evitare le trasmissioni di frame aggiuntivi che provocherebbero un aumento del consumo energetico. Le nuove versioni dello standard IEEE 802.15.4 sono state concepite per essere facilmente estendibili e, difatti, hanno previsto un campo del frame chiamato *Information Element* (IE). Quest'ultimo fornisce un modo semplice di incapsulare informazioni aggiuntive all'interno di un frame dati o di un frame di ACK. La struttura di un IE è composta da soli tre campi (lunghezza, ID e contenuto) e in un singolo frame è possibile concatenare molteplici IE. Dal momento che si conosce la loro lunghezza, tutti gli IE che hanno un ID non riconosciuto possono essere ignorati dal ricevitore. In questo modo è possibile estendere le funzionalità della modalità operativa TSCH senza avere problemi di compatibilità con le versioni precedenti. Infatti, i dispositivi che non riconoscono il comando di sleep possono continuare a funzionare normalmente, con l'unico svantaggio di non poter usufruire del risparmio energetico offerto da PRIL.

Per descrivere il funzionamento delle tecniche PRIL verrà usato il termine *link*, con riferimento all'insieme delle celle della matrice TSCH allocate per le trasmissioni tra un dispositivo e uno dei suoi vicini. Generalmente un link è formato da una sola cella ma, nei casi in cui l'overprovisioning è abilitato, esso include tutte le celle del bundle.

Il protocollo adottato nelle tecniche PRIL prevede che sia il trasmettitore, mediante l'uso del comando di sleep, a decidere quando e per quanto tempo il ricevitore debba disabilitare l'ascolto nelle celle a loro riservate. Quindi, in un dato momento, un link può essere attivo o inattivo. Quando è attivo, trasmettitore e ricevitore si comportano come previsto dal TSCH Standard e, pertanto, il ricevitore dovrà attivarsi in ogni cella del link per ascoltare il canale, in attesa di eventuali frame dati in arrivo. Invece, quando viene ricevuto il comando di sleep, il link viene disattivato

per il tempo indicato nel comando. In questo caso il nodo ricevitore non deve ascoltare il canale e, ovviamente, il nodo trasmettitore non può effettuare trasmissioni. Se vengono accodati nuovi frame dati nella coda del trasmettitore mentre il link è inattivo, questi potranno essere trasmessi solo una volta che il link verrà riattivato. Quindi, per ridurre in modo considerevole gli sprechi causati dal fenomeno di idle listening, il link deve rimanere inattivo il più a lungo possibile e tornare attivo solo per il tempo necessario alla trasmissione dei frame dati. Sfruttando appieno le caratteristiche del traffico, PRIL è in grado di prevedere quando sarà disponibile un nuovo frame dati da trasmettere. Solitamente la generazione dei pacchetti da parte delle applicazioni è periodica, quindi il calcolo del periodo di disattivazione è abbastanza semplice.

Per quanto riguarda la codifica del comando di sleep si hanno due possibilità. La durata del tempo di disattivazione del link può essere espressa sia come valore relativo sia come valore assoluto. Nel primo caso deve essere specificato il numero di ripetizioni della cella in cui il ricevitore non deve ascoltare il canale. Se viene utilizzato l'overprovisioning, tutte le celle del bundle devono essere conteggiate separatamente. Questo implica che il link possa essere inattivo per alcune celle del bundle e ritornare attivo per le rimanenti. Nel secondo caso viene usato l'ASN del timeslot in cui il nodo ricevitore deve iniziare ad ascoltare nuovamente il canale.

L'opzione migliore è quella di usare il numero relativo di celle in quanto permette di avere una rappresentazione compatta del comando di sleep. Il valore dell'ASN richiede 5 byte per essere rappresentato, mentre usando il numero di celle sono necessari solo pochi bit. Tuttavia, la scelta tra valore relativo e valore assoluto risulta essere equivalente in molte applicazioni pratiche in quanto il contributo energetico è spesso trascurabile se comparato all'energia necessaria per trasmettere l'intero frame dati e il relativo frame di ACK.

La prima tecnica PRIL sviluppata in [3] si chiama Proactive Reduction of Idle Listening at the First hop (PRIL-F) ed è specifica per il primo hop. PRIL-F ha effetto solo sul nodo che segue le sorgente nel percorso dei pacchetti verso la destinazione, il che lo rende ottimo per reti con pochi hop.

3.3 PRIL-F

PRIL-F offre un meccanismo che permette di ridurre l'energia sprecata in idle listening per i nodi al primo hop. Infatti, considerando la sequenza di nodi N_s, N_1, N_2, ..., N_d che rappresenta il percorso di un pacchetto dalla sorgente (N_s) verso la destinazione (N_d), PRIL-F ha effetto solo sul nodo N_1.

Il comando di sleep usato nella tecnica PRIL-F ha una struttura molto semplice, essendo formato solo dal campo T_{next} che indica il tempo in cui il link dovrà ritornare attivo. Per poter essere implementato nei dispositivi di una rete WSN, PRIL-F richiede solo che il nodo sorgente abbia la capacità di calcolare il tempo

T_{next} da includere nel comando di sleep. In molti casi questo requisito può essere facilmente soddisfatto. Spesso le applicazioni in esecuzione sui sensori sono programmate per effettuare rilevazioni e generare pacchetti periodicamente. Dato che il periodo di generazione è noto a priori, il calcolo di T_{next} è immediato. Inoltre, il valore di T_{next} calcolato da PRIL-F nei casi di traffico periodico permette di non influenzare le latenze di trasmissione rispetto al TSCH Standard e, allo stesso tempo, di ridurre drasticamente (se non proprio eliminare) il fenomeno di idle listening. Il trasmettitore, essendo a conoscenza del tempo in cui verrà generato il prossimo pacchetto, è in grado di scegliere un valore di T_{next} che permetta al nodo ricevitore di riattivarsi solo quando il nuovo frame dati sarà effettivamente pronto per essere trasmesso.

Tuttavia, PRIL-F può essere usato anche quando la generazione dei pacchetti da parte delle applicazioni è sporadica. In questo caso non c'è un periodo di generazione da cui partire per ricavare il valore di T_{next}. Per questo tipo di traffico, un valore troppo alto ridurrebbe senz'altro l'idle listening, ma avrebbe il difetto di aumentare le latenze di comunicazione, poiché l'applicazione potrebbe generare un pacchetto mentre il link è ancora inattivo. È possibile calcolare un valore di T_{next} che non peggiori le latenze di trasmissione solo se esiste un limite inferiore di tempo tra la generazione di un pacchetto e il successivo. In questo modo non si riesce ad eliminare completamente il consumo dovuto all'idle listening, ma si ha la certezza di non ritardare in modo consistente la consegna dei pacchetti. Usando il limite inferiore, tutte le volte che l'applicazione genera un pacchetto in ritardo rispetto al valore di T_{next} calcolato, si verifica idle listening. Ad ogni modo, finché è possibile calcolare un valore adeguato di T_{next}, PRIL-F permette di ridurre in modo consistente i consumi dovuti all'idle listening, pur mantenendo l'affidabilità e le latenze di comunicazione invariate rispetto al TSCH standard.

Entrando più nei dettagli, il funzionamento dell'algoritmo PRIL-F può essere schematizzato attraverso una macchina con due stati: *open* e *closed*. Quest'ultimi rappresentano lo stato di attivazione del link tra il nodo che genera i pacchetti (N_s) e il nodo che si trova al primo hop (N_{fh}) nel percorso dei pacchetti verso la destinazione. In particolare, il link può essere usato per trasmettere frame dati solo se è nello stato *open*. Al contrario, quando si trova nello stato *closed*, il link non può essere usato per comunicare.

In Figura 3.1 sono riportate sia la macchina a stati specifica per il nodo sorgente (N_s) sia quella per il nodo al primo hop (N_{fh}). Il funzionamento della macchina a stati del nodo ricevitore è quello più semplice. Quando il link è nello stato *open*, il nodo N_{fh} si comporta come nel TSCH standard e, quindi, si attiva in ogni cella del link per ascoltare il canale, in attesa di eventuali frame dati in trasmissione dal nodo N_s. Nelle celle in cui non viene trasmesso alcun frame si ha idle listening. Appena viene ricevuto un frame dati contenente il comando di sleep, il nodo N_{fh} disattiva il link che passa nello stato *closed* (arco *RX sleep*). In questa occasione viene estrapolato dall'information element il valore di T_{next}, il quale è usato per

inizializzare un contatore interno chiamato `sleep_end`. Quest'ultimo viene utilizzato per tenere traccia del numero di timeslot in cui il link deve rimanere ancora disabilitato ed è decrementato ad ogni ripetizione della cella $N_s \rightarrow N_{fh}$. Quando il contatore arriva a zero, si ha la transizione allo stato *open* (arco *sleep_end==0*) e il nodo N_{fh} può ritornare ad ascoltare il canale.

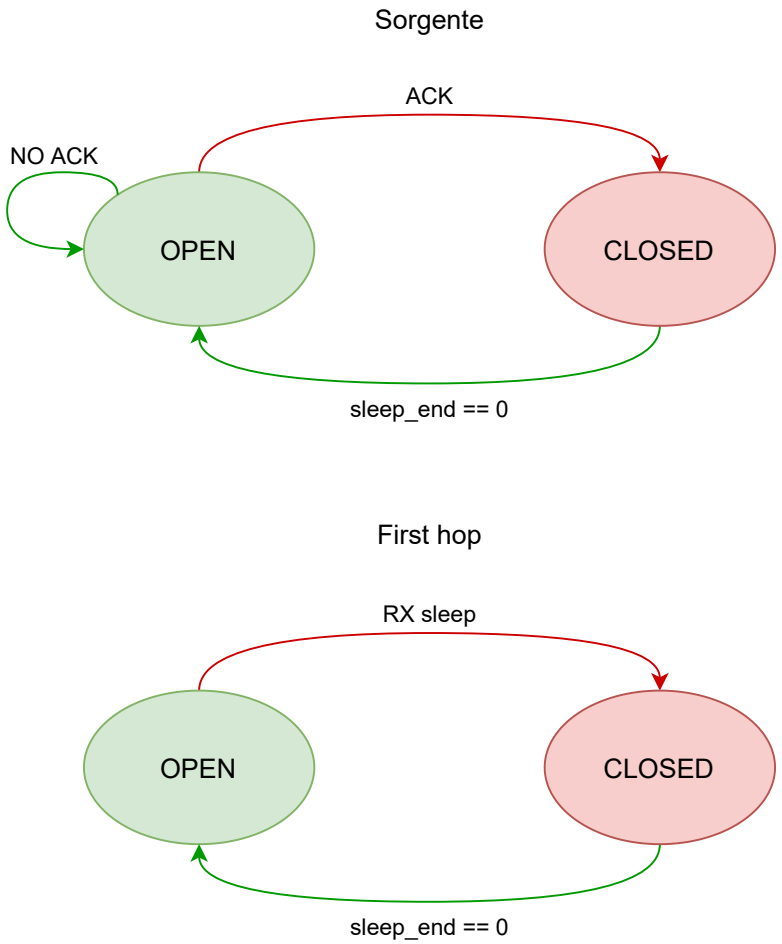


Figura 3.1: Macchine a stati del nodo trasmettitore (in alto) e del nodo ricevitore (in basso) in PRIL-F

La macchina a stati del nodo N_s è leggermente diversa in quanto bisogna prendere in considerazione l'eventualità che il frame dati contenente il comando di sleep non arrivi correttamente al nodo N_{fh} al primo tentativo. Inoltre, dal momento che i frame dati e i frame di ACK possono essere persi a causa delle interferenze sul canale di comunicazione usato per la trasmissione, è possibile che si verifichino situazioni in cui lo stato di attivazione del link dal lato del trasmettitore sia diverso dallo stato di attivazione del link dal lato del ricevitore. Infatti, a differenza del nodo N_{fh}, che disattiva il link alla ricezione del comando di sleep, il nodo N_s non

può disattivarlo subito dopo la trasmissione del frame dati contenente il comando, ma deve attendere la conferma di avvenuta ricezione dal nodo N_{fh}. Fintanto che il frame di ACK non è ricevuto, il link rimane nello stato *open* e, come previsto dal TSCH Standard, verranno ritentate le trasmissioni del frame dati. Se T_{next} è rappresentato usando un valore relativo, ad ogni ritrasmissione il nodo N_s deve anche aggiornare l'informazione contenuta nel comando di sleep. Qualora il frame dati continuasse ad essere perso, dopo aver raggiunto il numero massimo di ritrasmissioni consentite (N_{tries}), questo viene scartato. Di conseguenza, non avendo ricevuto il frame di ACK, il nodo N_s mantiene il link nello stato *open*.

In particolare, la mancata ricezione del frame di ACK è legata a due possibili scenari, che è utile analizzare per comprendere meglio il funzionamento della tecnica PRIL-F. Il primo si verifica quando il frame dati contenente il comando di sleep non riesce ad arrivare correttamente a destinazione in nessuno degli N_{tries} tentativi disponibili ed è lo scenario più sfavorevole per il nodo N_{fh} da un punto di vista energetico. Infatti, non avendo ricevuto il comando di sleep, il nodo N_{fh} mantiene il link nello stato *open* e non è in grado di impedire il fenomeno di idle listening. Al contrario, la seconda situazione è più svantaggiosa per il nodo N_s e si verifica quando è il frame di ACK ad essere perso. In questo caso lo stato di attivazione del link dal lato del nodo N_s e lo stato di attivazione del link dal lato del nodo N_{fh} differiscono. Infatti, il nodo N_{fh} riceve correttamente il comando di sleep e procede con la disattivazione del link. Invece, il nodo N_s, non avendo ricevuto il frame di ACK, mantiene il link attivo e tenta inutilmente le ritrasmissioni del frame dati. Maggiori dettagli su questo scenario sono riportati in Sezione 6.1 con riferimento alla Figura 6.2.

La transizione del link verso lo stato *closed* (arco *ACK*) ha luogo solo quando il nodo N_s riceve il frame di ACK dal nodo N_{fh}. In questa occasione, il nodo N_s inizializza il suo contatore interno `sleep_end` con l'ultimo valore di T_{next} trasmesso. Il contatore viene decrementato ad ogni ripetizione della cella $N_s \rightarrow N_{fh}$ e, quando il suo valore raggiunge lo zero, il link ritorna nello stato *open* (arco *sleep_end==0*). Se l'applicazione in esecuzione sul nodo N_s genera un pacchetto mentre il link è nello stato *closed*, questo potrà essere trasmesso solo quando il link sarà nuovamente nello stato *open*. Tuttavia, scegliendo il giusto valore di T_{next}, questa situazione non dovrebbe mai verificarsi.

In Figura 3.2 sono riportati alcuni esempi che mostrano il funzionamento della tecnica PRIL-F. Per semplicità, il numero massimo di tentativi per trasmettere un frame dati prima che questo sia scartato è $N_{tries} = 2$, ossia la trasmissione iniziale più una ritrasmissione. Il nodo N_s genera un nuovo pacchetto ogni $T_{app} = 10\,\text{s}$ e c'è una sola cella riservata alle comunicazioni tra N_s e N_{fh} che si ripete ogni 2 s. Quindi, un nuovo pacchetto è generato ogni 5 ripetizioni della cella $N_s \rightarrow N_{fh}$. In figura sono illustrate tre possibili situazioni (caso A, caso B e caso C), riportando per ognuna di esse lo stato di attivazione del link $N_s \rightarrow N_{fh}$ dal punto di vista del nodo ricevitore N_{fh} dallo slotframe 50 allo slotframe 55. Sotto ogni cella è indicato

anche l'esito della trasmissione del frame dati e del relativo frame di ACK. Inoltre, associato alla freccia relativa alla trasmissione del frame dati contenente il comando di sleep, è riportato anche il valore di T_{next} inviato. In questo esempio, il valore di T_{next} è stato codificato usando una rappresentazione relativa, che indica il numero di celle in cui il link deve rimanere inattivo.

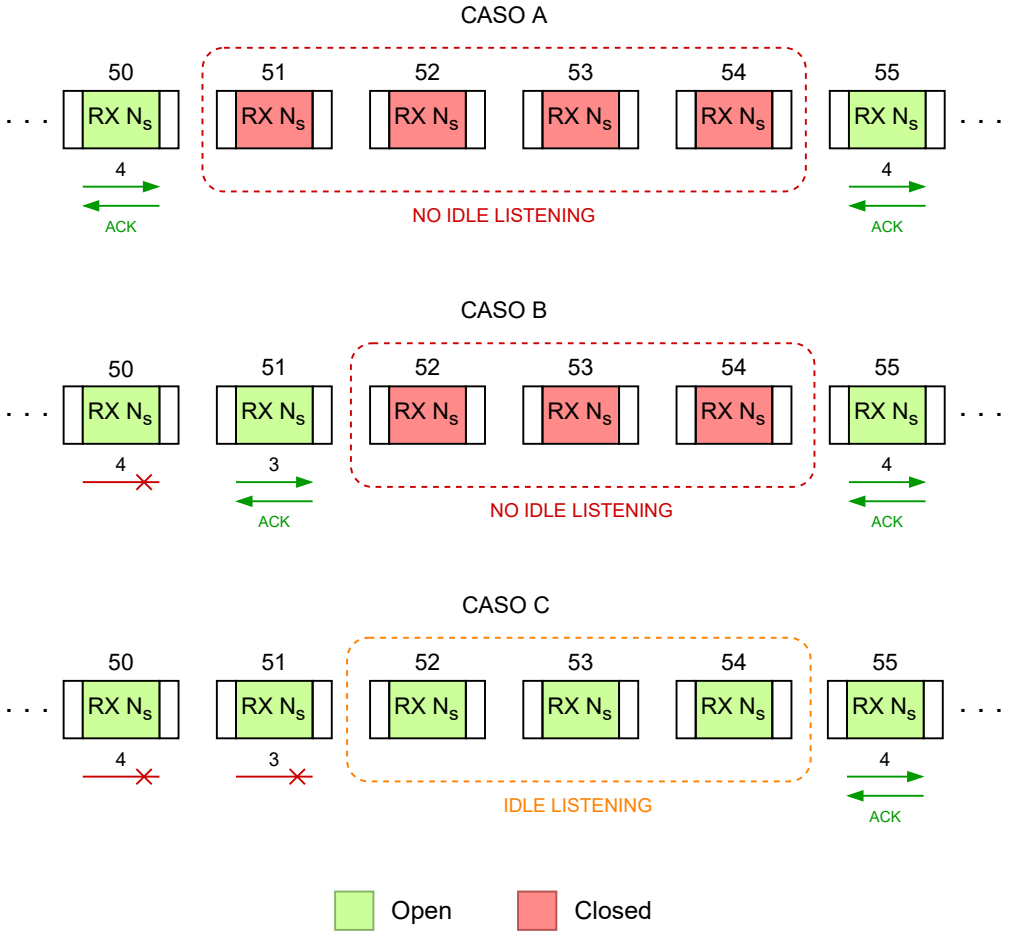


Figura 3.2: Esempi sul funzionamento della tecnica PRIL-F

Nel caso A, la trasmissione del frame dati avviene con successo al primo tentativo e il ricevitore può disabilitare la cella per le successive 4 ripetizioni, ossia fino allo slotframe 55 in cui il link ritorna attivo. Di conseguenza, il fenomeno di idle listening è stato evitato dallo slotframe 51 allo slotframe 54. Invece, nel caso B, il primo tentativo di trasmissione non va a buon fine e viene effettuata una ritrasmissione durante lo slotframe 51. Qui, sia il frame dati sia il frame di ACK arrivano correttamente a destinazione. Il link rimane nello stato *closed* dallo slotframe 52 allo slotframe 54, per poi tornare attivo nello slotframe 55. In questo caso è utile notare come il valore di T_{next} cambi nel secondo tentativo di trasmissione del frame

dati. Infatti, avendo usato un valore relativo, è necessario decrementare T_{next} ad ogni ritrasmissione. Questo è necessario per fare in modo che il link venga riattivato allo slotframe 55 anche in caso di ritrasmissioni in quanto, in tale slotframe, sarà presente un nuovo frame dati nella coda di N_s pronto per essere trasmesso. Se il valore non venisse decrementato, il link si riattiverebbe allo slotframe 56 e il frame dati subirebbe un ritardo sul tempo di consegna. Per quanto riguarda il caso C, la trasmissione del frame dati non ha successo nemmeno al secondo tentativo. Avendo scelto $N_{tries} = 2$, il frame viene scartato allo slotframe 51 e il nodo N_{fh} non riceve il comando di sleep. Quindi, il link rimane inutilmente attivo dallo slotframe 52 allo slotframe 54 e non si riesce ad evitare il fenomeno di idle listening.

Dagli esempi riportati in figura è facile vedere come, grazie alla tecnica PRIL-F, il consumo energetico del nodo al primo hop possa solo essere ridotto e mai incrementato. Infatti, nel peggiore dei casi (caso C), quando il frame dati contenente il comando di sleep viene perso, il link rimane nello stato *open* e non si hanno differenze rispetto al TSCH Standard. Invece, in tutti gli altri casi si ha un notevole beneficio in termini di energia risparmiata in quanto si riesce ad impedire al ricevitore di ascoltare il canale nelle celle in cui non ci saranno sicuramente frame nella coda del trasmettitore, eliminando così gli sprechi di energia dovuti all'idle listening.

Dal momento che PRIL-F ha effetto solo sui nodi che comunicano direttamente con la sorgete dei pacchetti, questa tecnica offre ottimi risultati solo se usata in reti con topologia a stella, oppure in reti con topologia ad albero in cui i nodi intermedi sono collegati ad un'alimentazione esterna e il traffico è formato solo da flussi di pacchetti che partono dalle foglie e arrivano alla radice. Spesso, però, i percorsi tra la sorgente e la destinazione sono formati da molti nodi e, in questi casi, il miglioramento energetico che si ottiene con PRIL-F è limitato. Infatti, nel caso di rete a più livelli, solo i nodi al primo hop possono beneficiare del risparmio energetico offerto da questa tecnica, mentre sui nodi intermedi nessun miglioramento può essere ottenuto.

Per poter ridurre i consumi energetici anche dei nodi intermedi, è stata ideata e sviluppata in questa tesi un'implementazione di PRIL che, a differenza di PRIL-F, sia in grado di funzionare oltre il primo hop. La tecnica è chiamata PRIL-MH e permette di propagare il comando di sleep in tutti i nodi che formano il percorso di un pacchetto.

Capitolo 4

PRIL-MH

4.1 Introduzione

La tecnica PRIL-MH è stata sviluppata per poter estendere il meccanismo di riduzione dell'idle listening in tutti i nodi di una WSN per cui sono allocate delle celle in ricezione. Infatti, come spiegato nel capitolo precedente, l'attuale implementazione di PRIL, PRIL-F, permette di ridurre i consumi solo sui nodi che si trovano al primo hop (sia nel percorso in salita sia nel percorso in discesa) e non porta nessun beneficio agli altri dispositivi. Ciononostante, considerando la riduzione dello spreco di energia dovuto all'idle listening, i risultati ottenuti nei dispositivi su cui PRIL-F agisce sono ottimi. Per questo motivo, PRIL-MH non è stata progettata per sostituire PRIL-F, bensì per cooperarci in modo da poter ottenere risultati migliori. Infatti, considerando la sequenza di nodi N_s, N_1, N_2, ..., N_d che rappresenta il percorso seguito da un pacchetto dal nodo sorgente (N_s) al nodo destinazione (N_d), PRIL-F agisce solo su N_1, mentre PRIL-MH ha effetto su tutti i nodi da N_2 a N_d.

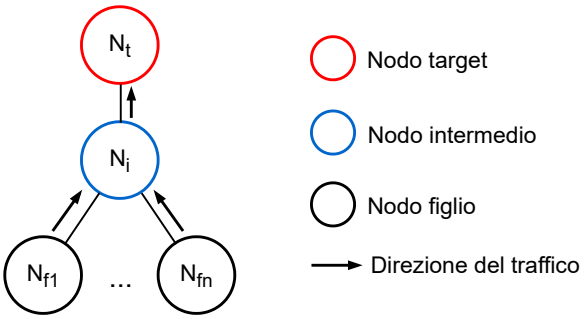


Figura 4.1: Visione della rete di un generico nodo intermedio in PRIL-MH

Per comprendere meglio il contesto in cui la tecnica PRIL-MH viene eseguita, si consideri la Figura 4.1 dove è riportata la visione della rete da parte di un generico

nodo intermedio (N_i) quando è in esecuzione PRIL-MH. In particolare, la tecnica mira a ridurre gli sprechi di energia causati dal fenomeno di idle listening sul nodo *target* (N_t), disattivando il link $N_i \rightarrow N_t$ sulla base dei periodi di generazione dei pacchetti ricevuti dai nodi figli (N_{f1}, ..., N_{fn}). Infatti, a differenza di PRIL-F, il link su cui PRIL-MH agisce può essere attraversato da molteplici flussi di pacchetti, dove per flusso si intende il traffico dati generato da un nodo sorgente verso lo stesso nodo destinazione.

Quindi, per poter decidere quando è meglio disattivare il link verso il target, un nodo dovrebbe tenere conto dell'arrivo previsto dei pacchetti di tutti i flussi che usano tale link per raggiungere la destinazione. Purtroppo, a causa delle risorse limitate di cui dispongono i dispositivi usati nelle WSN e dalla impredicibilità degli accodamenti di pacchetti sui nodi intermedi, un algoritmo di questo tipo è difficilmente realizzabile. Basti pensare a reti in cui sono presenti tanti nodi sorgente che generano pacchetti. Man mano che ci si avvicina alla radice, tutti i flussi si riuniscono in pochi link e i dispositivi si troverebbero a dover gestire il tempo di arrivo di un numero esorbitante di pacchetti.

La soluzione ideata per PRIL-MH prevede che il link si disattivi solo sulla base del tempo di arrivo previsto dei pacchetti del flusso caratterizzato dal periodo di generazione più breve. Infatti, quest'ultimo è solitamente quello ad avere maggiori vincoli sul tempo di consegna. Di conseguenza, prima di poter iniziare a trasmettere il comando di sleep al nodo target, un nodo intermedio ha bisogno di conoscere i flussi che attraversano tale link e trovare quello con il periodo di generazione minimo. Per questo motivo, il nodo su cui PRIL-MH è in esecuzione evolve tra due fasi:

- **Learning**: il nodo apprende i periodi di generazione dei flussi destinati al nodo target e ne calcola il minimo (P_{min}).
- **Runtime**: il nodo ha appreso il periodo di generazione minimo (P_{min}) e può iniziare a trasmettere il comando di sleep al nodo target per ridurre il fenomeno di idle listening.

La tecnica PRIL-MH è stata sviluppata in tre versioni: PRIL-MH Base (PRIL-MHB), PRIL-MH Improved (PRIL-MHI) e PRIL-MH Improved 2 (PRIL-MHI2). Tuttavia, la fase di *learning* è rimasta uguale tra una versione e l'altra, mentre quello che differisce è la fase di *runtime*.

4.2 Learning

Durante la fase di *learning*, per ogni nodo target (N_t), il nodo N_i deve apprendere qual è il flusso con periodo di generazione minimo (P_{min}) tra quelli che sfruttano il link $N_i \rightarrow N_t$ per raggiungere la destinazione. Dal momento che N_i non ha modo

di risalire al periodo di generazione di un flusso, in PRIL-MH è stato previsto che tutti i nodi sorgente aggiungano in ogni frame dati il periodo con cui l'applicazione genera un nuovo pacchetto (T_{app}). Così come si fa per il comando di sleep, anche per trasmettere il valore di T_{app} viene usato l'information element.

Quindi, alla ricezione del primo frame dati da inoltrare a N_t, il nodo N_i estrae il valore di T_{app} dall'information element e ha inizio la fase di *learning* per il link $N_i \rightarrow N_t$, la quale finirà solo quando sarà passato un periodo di tempo pari al valore di T_{app} appena ricevuto. Da questo momento, ogni volta che viene ricevuto un nuovo frame dati da trasmettere a N_t, il nodo N_i, utilizzando il valore contenuto nel campo T_{app}, apprende il periodo di generazione del flusso. Trascorso il periodo di tempo prestabilito, la fase di *learning* può terminare e si passa alla fase di *runtime*. Di tutti i periodi di generazione appresi fino a quel momento, il nodo N_i ne calcola il minimo e usa tale valore come periodo minimo di riferimento (P_{min}) per la disattivazione del link $N_i \rightarrow N_t$.

Ci sono situazioni in cui uno stesso link è attraversato da più flussi con periodo di generazione P_{min}. Considerare equivalenti, ai fini dell'algoritmo, i pacchetti con periodo minimo di flussi diversi potrebbe causare problemi nella fase di *runtime*. Per questo motivo, il nodo N_i sceglie uno tra i nodi sorgente di tali flussi da usare come nodo riferimento (N_{ref}). In particolare, viene scelto il nodo sorgente del flusso relativo al primo pacchetto con periodo P_{min} accodato in N_i. In fase di *runtime*, per disattivare il link, l'algoritmo prenderà in considerazione solo il tempo di accodamento dei pacchetti con periodo P_{min} generati dal nodo N_{ref}.

Dal momento che la disattivazione del link può avvenire solo una volta appreso il valore di P_{min}, N_i non può trasmettere il comando di sleep a N_t durante la fase di *learning* e, pertanto, i due nodi si comportano come previsto dal TSCH Standard. Di conseguenza, in ogni cella in cui il trasmettitore (N_i) non ha frame in coda da inviare, sul nodo ricevitore (N_t) si ha il fenomeno di idle listening. Per fare in modo che lo spreco di energia sia limitato, è necessario che la fase di *learning* duri il meno possibile per passare alla fase di *runtime*.

Tuttavia, una volta passati alla fase di *runtime* possono verificarsi delle situazioni che potrebbero causare un ritorno nella fase di *learning*. Infatti, se il nodo N_i non riceve più i pacchetti generati dal nodo N_{ref}, non può essere innescato il processo che permette al nodo target di disattivare l'ascolto del canale ed è, quindi, necessario cominciare una nuova fase di *learning* per apprendere dei nuovi valori di P_{min} e N_{ref}. Ad esempio, questo può accadere a causa della presenza di ostacoli che provocano la perdita di tutti i pacchetti generati dal nodo N_{ref} prima che questi possano raggiungere il nodo N_i. La mancata ricezione dei pacchetti potrebbe anche essere dovuta allo spostamento del nodo N_{ref} in un altro punto della rete. Infatti, questo porta a dei cambiamenti nel percorso seguito dai pacchetti che potrebbe non includere più il link $N_i \rightarrow N_t$.

Da un punto di vista energetico, avviare una nuova fase di *learning* è un evento negativo in quanto, in questa fase, non è possibile in alcun modo agire per ridurre

lo spreco dovuto al fenomeno di idle listening. Per poter rimanere in *runtime* anche a seguito della mancata ricezione dei pacchetti da N_{ref}, durante la fase di *learning* il nodo N_s memorizza dei valori di riserva (P_{backup} e N_{backup}) da usare quando quelli principali non sono più validi. In particolare, viene memorizzato il secondo periodo più piccolo (P_{backup}) immediatamente successivo al valore di P_{min} e il relativo nodo di riferimento (N_{backup}). Questo meccanismo permette di evitare il passaggio alla fase di *learning* semplicemente utilizzando P_{backup} e N_{backup} come nuovi valori di P_{min} e N_{ref}. Qualora la stessa situazione dovesse verificarsi anche per i pacchetti relativi al nodo N_{backup}, il ritorno alla fase di *learning* non potrebbe essere evitato.

4.3 PRIL-MHB

PRIL-MHB è la prima versione sviluppata della tecnica PRIL-MH e permette di ridurre i consumi legati al fenomeno di idle listening su tutti i nodi che costituiscono il percorso di un pacchetto, ad eccezione dei nodi al primo hop in cui è stata mantenuta PRIL-F.

La struttura del comando di sleep utilizzata nella tecnica PRIL-MHB è rimasta molto semplice in quanto è presente un solo campo (T_{end}) che indica al nodo target fino a quando il link deve rimanere inattivo. Anche in questo caso, per rappresentare il valore di T_{end} può essere usata sia una codifica relativa sia una codifica assoluta. Tuttavia, dal momento che in ogni frame dati deve anche essere aggiunto il periodo di generazione T_{app}, la rappresentazione relativa è da preferirsi per evitare di aumentare in modo considerevole la dimensione dei frame.

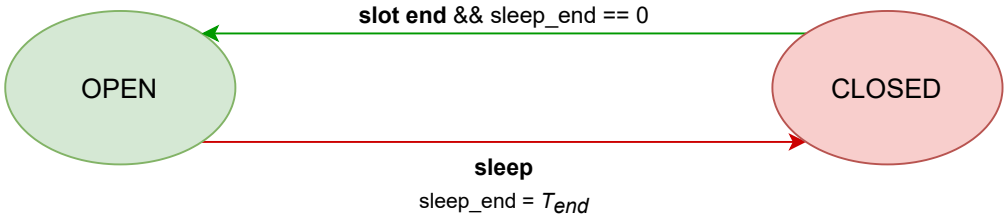


Figura 4.2: Macchina a stati del nodo ricevitore in PRIL-MHB

Entrando nel merito della tecnica, il funzionamento di PRIL-MHB nella fase di *runtime* può essere sintetizzato mediante una macchina a stati. In Figura 4.2 è stata riportata la macchina a stati del nodo ricevitore (N_t) e in Figura 4.3 è riportata quella del nodo trasmettitore (N_i). Su ogni arco, prima della condizione che deve essere verificata per poter effettuare la transizione da uno stato all'altro, è specificato anche l'evento a seguito del quale la transizione può avvenire. In particolare, si hanno quattro tipi di eventi: *sleep*, che indica la ricezione del comando di sleep, *ack*, che indica la corretta ricezione del frame di ACK, *no ack*, che indica la mancata ricezione del frame di ACK e *slot end*, che indica la fine della cella

$N_i \rightarrow N_t$. Inoltre, per rendere maggiormente chiaro l'algoritmo, dove necessario sono state inserite parti rilevanti del codice associato alle transizioni.

La macchina a stati del ricevitore è rimasta uguale a quella della tecnica PRIL-F. Anche qui, fino a quando il link è nello stato *open*, il nodo N_t si comporta come nel TSCH Standard. Alla ricezione del comando di sleep, N_t estrae dall'*information element* il valore di T_{end} e lo usa per inizializzare il suo contatore interno `sleep_end`. Quindi, il link viene disattivato e passa allo stato *closed*. Da questo momento, ad ogni ripetizione della cella $N_i \rightarrow N_t$, il contatore viene decrementato. Quando `sleep_end` arriva a zero, il link ritorna alla stato *open* e il nodo N_t può riabilitare l'ascolto del canale.

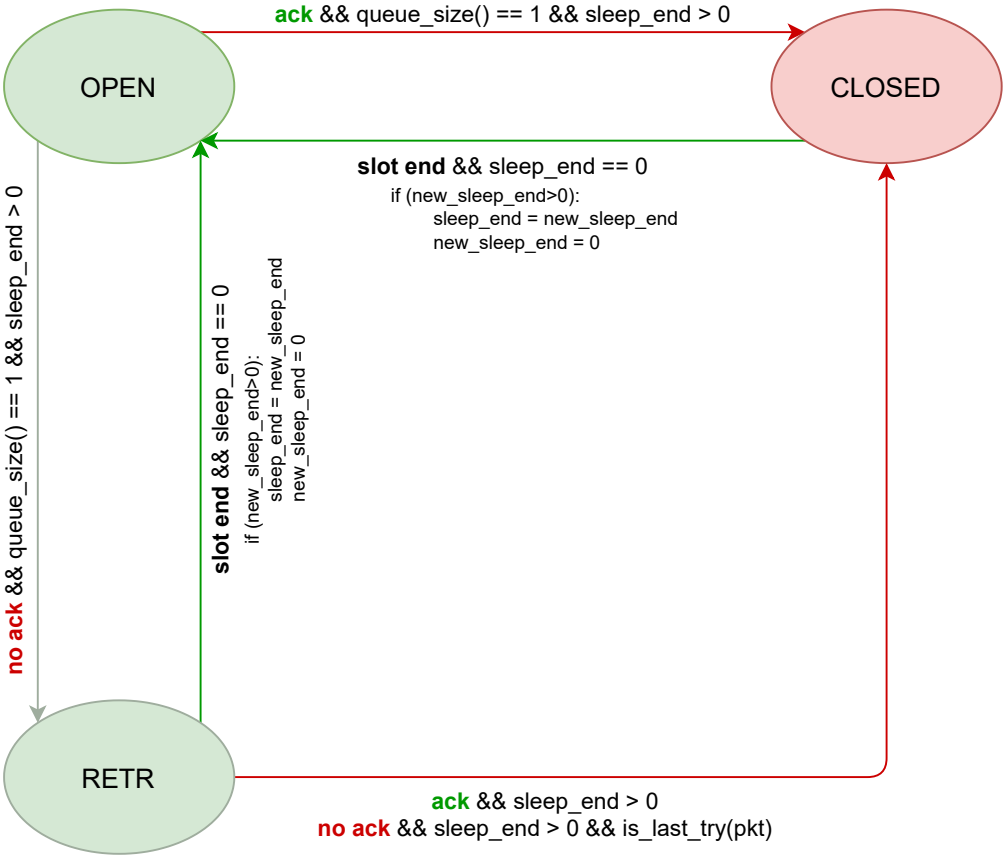


Figura 4.3: Macchina a stati del nodo trasmettitore in PRIL-MHB

La macchina a stati del trasmettitore presenta delle differenze sostanziali rispetto a quella della tecnica PRIL-F. Alla conclusione della fase di *learning*, il nodo N_i ha appreso i valori di P_{min} e N_{ref}, e può iniziare la fase di *runtime*. Il link inizialmente si trova nello stato *open*. Quando nel nodo trasmettitore viene accodato il pacchetto con periodo P_{min} generato dal nodo N_{ref}, il meccanismo volto alla disattivazione temporanea del link può avere inizio. In particolare, in questa

occasione il nodo N_i inizializza il suo contatore interno `sleep_end` calcolando, a partire dal periodo P_{min} e dalla configurazione della matrice TSCH, il numero di celle $N_i \rightarrow N_t$ che si ripeteranno prima dell'arrivo previsto del prossimo pacchetto con periodo minimo. Da questo momento il contatore viene decrementato a ogni ripetizione della cella $N_i \rightarrow N_t$.

Nonostante il contatore `sleep_end` sia stato inizializzato, il link rimane nello stato *open*. Infatti, l'inizializzazione serve solo a tenere traccia del tempo in cui è prevista la ricezione del prossimo pacchetto con periodo minimo. È bene precisare che PRIL-MHB è eseguito sui nodi intermedi di una rete, i quali possono ricevere pacchetti da più nodi figli da inoltrare verso lo stesso nodo target. Di conseguenza, è possibile che nella coda di N_i non sia presente solo il pacchetto con periodo minimo, ma anche quelli accodati precedentemente e non ancora trasmessi a N_t. Inoltre, prima che il pacchetto con periodo minimo sia trasmesso, N_i potrebbe ricevere ulteriori pacchetti da recapitare al nodo target. Quindi, per evitare che il link venga disattivato mentre in coda sono presenti ancora dei pacchetti, il comando di sleep può essere aggiunto solo durante la trasmissione dell'ultimo pacchetto che, in alcuni casi, potrebbe non coincidere con il pacchetto con periodo minimo.

Durante la trasmissione dell'ultimo frame dati in coda, se il valore del contatore è ancora maggiore di zero, N_i aggiunge il comando di sleep e trasmette il frame dati, usando il valore di `sleep_end` per il calcolo di T_{end}. Se la trasmissione va a buon fine al primo tentativo, ossia viene ricevuto il frame di ACK, il link passa direttamente allo stato *closed*. Altrimenti, nel caso in cui il primo tentativo sia fallimentare, il link passa allo stato *retr*. Questo stato è stato introdotto proprio per gestire le ritrasmissioni del frame dati contenente il comando di sleep. Quando è in *retr*, il link dal lato del nodo N_i continua ad essere attivo, ma non si ha nessuna informazione sullo stato di attivazione sul nodo N_t. Quindi, fino a quando non viene ricevuto un frame di ACK, il nodo N_i ritenta la trasmissione del frame dati. Se in uno dei tentativi la trasmissione ha successo e viene ricevuto il frame di ACK, il link passa allo stato *closed* e N_i ha la certezza che questo verrà disattivato anche da N_t. Invece, se il frame dati continua ad essere perso, dopo aver effettuato tutti gli N_{tries} tentativi disponibili, il frame viene scartato. In questo caso si ha ugualmente la transizione verso lo stato *closed*, però, il nodo N_i non ha la sicurezza che il nodo target abbia effettivamente ricevuto il comando di sleep e disattivato il link. Tuttavia, per evitare di perdere ulteriori frame e peggiorare l'affidabilità della rete, il nodo N_i considera il link dal lato del ricevitore inattivo. Infatti, N_t potrebbe aver ricevuto il comando di sleep e, se si decidesse di mantenere il link attivo, N_i trasmetterebbe eventuali nuovi frame dati in coda. Di conseguenza, tutte le trasmissioni effettuate mentre il link sul nodo target è disabilitato non andrebbero a buon fine, e i pacchetti verrebbero scartati prima che questi possano raggiungere la destinazione. Questi pacchetti avrebbero a priori una minore probabilità di giungere a destinazione rispetto ad altri in quanto il loro numero massimo di trasmissioni sarebbe inferiore a N_{tries}. Invece, disabilitando il link, N_i dovrà aspettare la prossima riattivazione prima di

poter trasmettere eventuali frame in coda.

Una volta che il link è nello stato *closed*, il nodo N_i attende che il contatore `sleep_end` arrivi a zero. Raggiunto lo zero, il link passa dallo stato *closed* allo stato *open*. Seppur meno frequente, è possibile che il link ritorni allo stato *open* senza passare per lo stato *closed*. Questo accade quando la trasmissione dell'ultimo frame dati in coda avviene quando il contatore `sleep_end` è già prossimo allo zero e, a causa delle ritrasmissioni, raggiunge lo zero quando il link è ancora nello stato *retr*.

Inoltre, sempre con riferimento alla macchina a stati del nodo N_i, è possibile notare che sotto agli archi che permetto la transizione verso lo stato *open* è stato riportato un frammento di codice in cui compare la variabile `new_sleep_end`. Quest'ultima rappresenta un contatore utilizzato dal nodo N_i per gestire l'arrivo del pacchetto con periodo P_{min} quando il link si trova nello stato *retr* o nello stato *closed*. Infatti, può succedere che il pacchetto arrivi nella coda del nodo N_i prima del previsto. In particolare, se nel periodo precedente il pacchetto con periodo minimo ha subito delle ritrasmissioni o degli accodamenti, quello successivo potrebbe arrivare nella coda del nodo N_i prima della riattivazione del link. Dal momento che, quando il link si trova in *retr* o in *closed*, il contatore `sleep_end` è sicuramente già in uso, il suo valore non può essere sovrascritto. Pertanto, al suo posto viene temporaneamente utilizzato `new_sleep_end` per tenere traccia del tempo di riattivazione da utilizzare nel periodo successivo. Come accade per `sleep_end`, anche `new_sleep_end` deve essere decrementato ad ogni ripetizione della cella $N_i \rightarrow N_t$. Una volta che il link ritorna nello stato *open*, il contatore `sleep_end` può essere riutilizzato e, quindi, gli viene assegnato l'attuale valore di `new_sleep_end` e l'intero processo descritto finora può ripetersi.

Al fine di comprendere meglio il funzionamento della tecnica PRIL-MHB, in Figura 4.4 sono illustrate quattro possibili situazioni (caso A, caso B, caso C e caso D) e, per ognuna di esse, è stato riportato sia lo stato di attivazione del link dal lato del trasmettitore (N_i) sia lo stato di attivazione del link dal lato del ricevitore (N_t). Per evidenziare meglio il comportamento della tecnica, il numero massimo di tentativi per trasmettere un frame prima che questo sia scartato è stato impostato a $N_{tries} = 2$. Nella matrice TSCH è allocata una sola cella $N_i \rightarrow N_t$ che si ripete ogni 2 s.

Tutti e quattro gli scenari illustrati si sviluppano a partire dalla stessa situazione iniziale in cui, durante lo slotframe 50, nel nodo N_i vengono accodati due pacchetti: P_1 e P_2. Il pacchetto P_1 appartiene al flusso con periodo di generazione minimo $P_{min} = 10$ s e, a seguito del suo accodamento nel nodo N_i, il contatore `sleep_end` viene inizializzato. Dato che la cella si ripete ogni 2 s e che il periodo di generazione è $T_{app} = 10$ s, la ricezione del prossimo pacchetto con periodo minimo è prevista per lo slotframe 55. Di conseguenza, il contatore `sleep_end` viene inizializzato con il valore 4, che sta a significare che non è previsto l'arrivo di ulteriori pacchetti con periodo minimo per le successive 4 ripetizioni della cella $N_i \rightarrow N_t$.

Figura 4.4: Esempi sul funzionamento della tecnica PRIL-MHB

Analizzando singolarmente le quattro situazioni, si ha che nel caso A il pacchetto P_1 e il pacchetto P_2 vengono trasmessi entrambi correttamente al primo tentativo. Quindi, durante lo slotframe 51, dato che nella coda del nodo N_i è presente solo il pacchetto P_2, viene aggiunto il comando di sleep al frame dati in trasmissione. Di conseguenza, ricevitore e trasmettitore possono disattivare il link a partire dallo slotframe successivo, impedendo il fenomeno di idle listening dallo slotframe 52 allo slotframe 54.

Per quanto riguarda il caso B, la trasmissione del pacchetto P_2, e quindi del comando di sleep, non ha successo al primo tentativo e il link sul nodo N_i passa allo stato *retr*. Quindi, durante lo slotframe 52 viene ritentata la trasmissione che, questa volta, va a buon fine e permette di evitare l'idle listening negli slotframe 53 e 54.

Al contrario, nel caso C neanche il secondo tentativo ha successo e, dato che il massimo numero di trasmissioni è $N_{tries} = 2$, il pacchetto P_2 viene scartato. Quindi, non avendo ricevuto il comando di sleep, N_t mantiene il link attivo e non si riesce a impedire il fenomeno di idle listening. Tuttavia, pur non avendo ricevuto il frame di ACK relativo al frame dati contenente il comando di sleep, il nodo N_i disattiva il link. Come accennato in precedenza, a seguito della perdita del pacchetto con il comando di sleep, il trasmettitore non ha nessuna informazione sullo stato di attivazione del link dal lato del ricevitore. Quindi, per evitare di perdere eventuali frame accodati successivamente, il link viene considerato inattivo fino a quando `sleep_end` non raggiunge lo zero. In questo modo, anche se N_t avesse ricevuto il comando di sleep, il link sarebbe comunque già attivo prima che N_i possa ricominciare a trasmettere i frame dati.

Per maggiore chiarezza, l'eventualità appena descritta è illustrata nel caso D in cui il frame dati contenente il comando di sleep arriva correttamente a destinazione al primo tentativo, ma il frame di ACK viene perso. Quindi, il ricevitore disattiva il link a partire dallo slotframe 52, mentre sul nodo N_i il link passa allo stato *retr* e il dispositivo tenta inutilmente la ritrasmissione. Come nel caso precedente, il frame viene scartato alla fine dello slotframe 52 e il link dal lato del trasmettitore passa allo stato *closed*. In questo caso, però, lo stato di attivazione del link non differisce tra trasmettitore e ricevitore. Da quest'ultimo esempio risulta chiaro che, se il link non venisse disattivato a seguito degli N_{tries} tentativi, eventuali frame dati accodati nel nodo N_i verrebbero trasmessi nonostante il ricevitore abbia disattivato il link e, pertanto, andrebbero persi, peggiorando l'affidabilità della rete.

Nonostante PRIL-MHB permetta di ridurre in modo considerevole lo spreco di energia dovuto all'idle listening, allo stesso tempo provoca un aumento della latenza di comunicazione. Infatti, disattivando il link sulla base dell'arrivo del pacchetto con periodo di generazione minimo, si ha un rallentamento su tutti i pacchetti degli altri flussi. Quest'ultimi possono trovarsi bloccati in un nodo intermedio in attesa che il link verso il target torni nuovamente attivo. In PRIL-MHB tutti i pacchetti in coda che si trovano dopo il pacchetto con periodo minimo rimangono sempre

bloccati nell'hop successivo.

In Figura 4.5 è riportato uno esempio che illustra la situazione sopracitata. Si consideri la sequenza di nodi N_f, N_i, N_t che rappresenta la parte di percorso che tre flussi diversi hanno in comune. Durante lo slotframe 100, nel nodo N_f vengono accodati in ordine P_2, P_1 e P_3, dove P_1 è il pacchetto appartenente al flusso con periodo di generazione minimo. Nello stesso slotframe, sia il link $N_f \rightarrow N_i$ sia il link $N_i \rightarrow N_t$ sono attivi e, quindi, il pacchetto P_2 raggiunge subito il nodo N_t. Successivamente, nello slotframe 101, il nodo N_f trasmette a N_i anche il pacchetto P_1 che, essendo quello con periodo minimo, fa si che il nodo N_i imposti il suo contatore interno `sleep_end` e attivi il processo che porta alla disattivazione del link $N_i \rightarrow N_t$. Infatti, dal momento che ha solo un pacchetto in coda, durante la trasmissione di P_1 il nodo N_i aggiunge il comando di sleep per disattivare il link. Quindi, quando nello slotframe 102 N_i riceve da N_f il pacchetto P_3, il link $N_i \rightarrow N_t$ è già inattivo e il pacchetto rimane bloccato in coda fino alla sua prossima riattivazione.

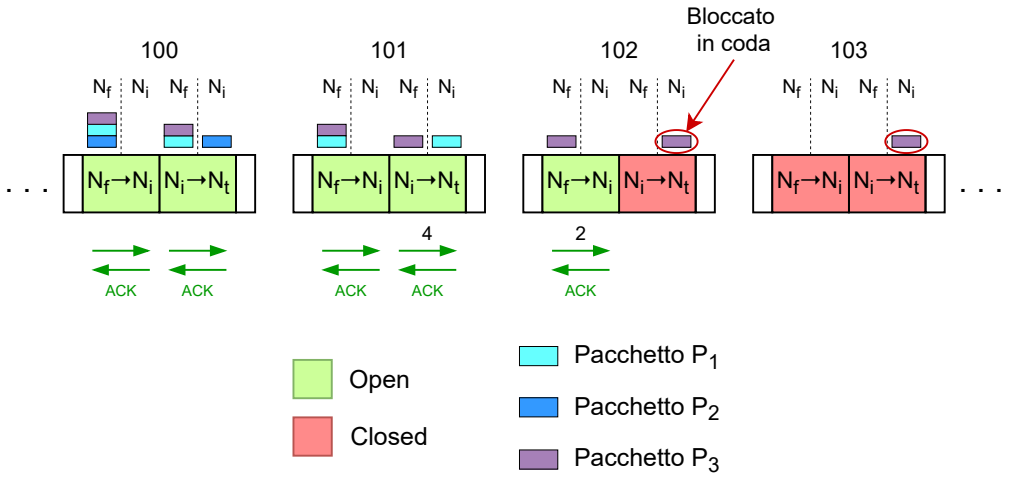


Figura 4.5: Esempio riguardo l'aumento delle latenze in PRIL-MHB

Dall'esempio illustrato è evidente che questa situazione provoca un aumento considerevole delle latenze e si verifica per ogni pacchetto in coda dopo quello con periodo minimo. Tuttavia, tale situazione potrebbe essere evitata se il ricevitore posticipasse la disattivazione del link qualora nella coda del trasmettitore ci fossero ancora dei frame dati pronti per essere trasmessi. Sulla base di questa idea è stata sviluppata una versione migliorata di PRIL-MHB, chiamata PRIL-MHI.

4.4 PRIL-MHI

PRIL-MHI è una versione della tecnica PRIL-MH sviluppata con l'obiettivo di risolvere uno specifico problema della tecnica PRIL-MHB che causa un aumento delle latenze di comunicazione. Infatti, usando PRIL-MHB, è possibile che si verifichino delle situazioni in cui il nodo N_i disattiva il link verso il target nonostante i nodi figli abbiano ancora dei pacchetti in coda da trasmettere. Per superare questa limitazione, ci si è concentrati sullo sviluppo di un algoritmo che fosse in grado di posticipare la disattivazione del link sulla base delle informazioni che il nodo N_i riceve dai nodi figli. A tal proposito, ogni nodo aggiunge nell'information element dei frame dati in trasmissione il numero di pacchetti che ha ancora in coda da trasmettere.

Una volta presa la decisione su quanto posticipare la disattivazione del link, il ricevitore viene informato mediante l'utilizzo di un nuovo campo del comando di sleep, chiamato T_{start}. Quindi, la coppia T_{start} e T_{end} indicano al nodo target rispettivamente quando inizia e finisce il periodo in cui il link $N_i \rightarrow N_t$ deve rimanere inattivo.

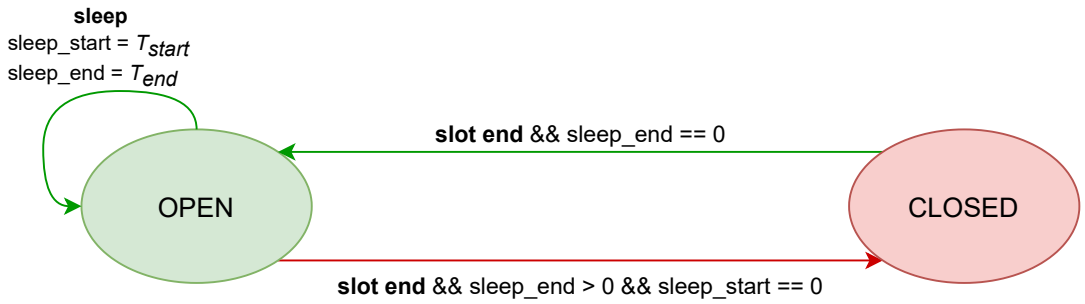


Figura 4.6: Macchina a stati del nodo ricevitore in PRIL-MHI

La macchina a stati del nodo target (Figura 4.6) è stata modificata rispetto a PRIL-MHB. Innanzitutto, è stato aggiunto un nuovo contatore, `sleep_start`, che indica per quanto tempo il link deve rimanere ancora attivo dopo la ricezione del comando di sleep. Il contatore viene inizializzato a partire dal valore del campo T_{start} ed è decrementato ad ogni ripetizione della cella $N_i \rightarrow N_t$. Quindi, a differenza di quanto accade in PRIL-MHB, il ricevitore non passa allo stato *closed* subito dopo la ricezione del comando di sleep, ma deve attendere che il contatore `sleep_start` arrivi a zero.

Per quanto riguarda il trasmettitore (N_i), la macchina a stati (Figura 4.7) è stata ulteriormente estesa. A differenza del nodo target, il trasmettitore ha bisogno di tre nuovi contatori per gestire l'inizio del periodo di disattivazione del link: `sleep_start`, `new_sleep_start` e `min_sleep_start`. Il contatore `sleep_start` indica il tempo in cui il link dovrà essere disattivato. Il contatore `new_sleep_start` ha la stessa utilità di `new_sleep_end` per `sleep_end`, ossia serve a memorizzare

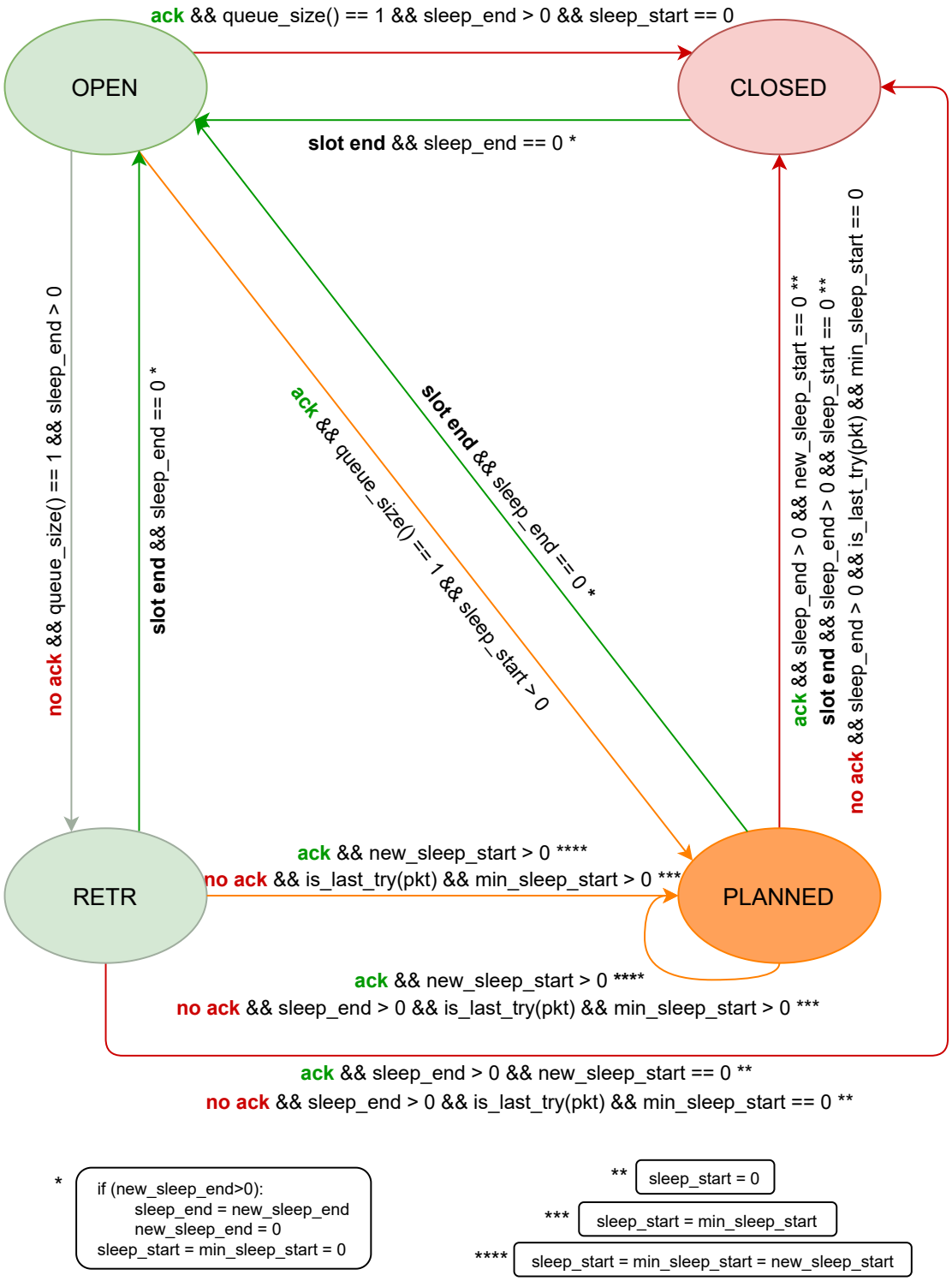


Figura 4.7: Macchina a stati del nodo trasmettitore in PRIL-MHI

temporaneamente un nuovo valore di `sleep_start` mentre l'attuale valore del contatore principale non può essere sovrascritto. Infine, `min_sleep_start` mantiene il più piccolo valore di T_{start} trasmesso ed è utilizzato per prendere decisioni quando un frame dati contenente il comando di sleep viene perso e non si hanno informazioni sicure (cioè confermate da un frame di ACK) sullo stato di attivazione del link dal lato del ricevitore. Tutti e tre i contatori vengono decrementati ad ogni ripetizione della cella $N_i \rightarrow N_t$.

Posticipare la disattivazione del link richiede particolare attenzione da parte del trasmettitore. Per questo motivo, nella macchina a stati è stato introdotto il nuovo stato *planned* che si occupa esclusivamente di gestire l'inizio del periodo di disattivazione. Infatti, quest'ultimo è deciso sulla base dello stato attuale della coda dei nodi figli, che è un'informazione dinamica che varia nel tempo. Di conseguenza, anche il valore di `sleep_start` potrebbe cambiare tra una trasmissione e la successiva per rispecchiare l'attuale situazione delle code.

Per descrivere il comportamento del trasmettitore nella tecnica PRIL-MHI, si consideri la macchina a stati a partire dallo stato *open*. In questo stato, così come accade anche in PRIL-MHB, il nodo N_i può aggiungere il comando di sleep solo all'ultimo frame dati in coda e solo dopo aver ricevuto il pacchetto con periodo minimo. Durante la trasmissione dell'ultimo frame in coda, N_i calcola un valore per T_{start} sulla base dell'attuale situazione delle code e utilizza tale valore per inizializzare anche il contatore `sleep_start`. Il valore zero indica che non c'è la necessità di mantenere il link ancora attivo, che può, quindi, essere disattivato subito. Pertanto, quando viene trasmesso un comando di sleep con un valore di T_{start} pari a zero, alla ricezione del frame di ACK il link passa direttamente allo stato *closed* e viene disattivato. Invece, se il valore di T_{start} trasmesso è maggiore di zero, alla ricezione del frame di ACK si ha la transizione verso lo stato *planned*.

In questo stato il link è ancora da considerarsi attivo, ma solo temporaneamente. Il contatore `sleep_start` indica il numero di celle $N_i \rightarrow N_t$ che il nodo N_i ha a disposizione per trasmettere eventuali frame dati prima che il link sul nodo N_t venga disattivato. Tuttavia, il tempo di inizio della disattivazione del link può essere modificato. Ad ogni trasmissione viene ricalcolato un nuovo valore di T_{start} che tiene conto dello stato attuale delle code. In particolare, sulla base delle nuove informazioni ricevute dai nodi figli e sullo stato attuale della coda del nodo N_i, è possibile posticipare o anticipare la disattivazione del link rispetto all'attuale tempo di inizio impostato. Il valore di `sleep_start`, però, non può essere sovrascritto subito. Infatti, il contatore è già in uso per tenere traccia dell'ultimo valore di T_{start} sicuramente ricevuto da N_t, ossia l'ultimo valore confermato dalla ricezione del frame di ACK relativo al frame dati che lo conteneva. Quindi, per memorizzare il nuovo valore di T_{start} trasmesso viene usato il contatore `new_sleep_start` e solo all'avvenuta ricezione del frame di ACK tale valore diventa effettivo e può essere riportato in `sleep_start`. Fintanto che `new_sleep_start` è inizializzato con un valore maggiore di zero, il link rimane nello stato *planned*. Al contrario, se il valore

di `new_sleep_start` è uguale a zero, alla ricezione del frame di ACK il link viene disattivato e passa allo stato *closed*.

Ci sono altre due situazioni che portano il link a passare dallo stato *planned* allo stato *closed*. La situazione più comune è quella in cui il contatore `sleep_start` raggiunge lo zero. Solitamente, questo accade quando il link è stato lasciato attivo, ma nessun frame è stato accodato in N_i nel frattempo. Invece, l'altra situazione si verifica quando il frame dati, dopo N_{tries} ritrasmissioni, viene perso e il contatore `min_sleep_start` è uguale a zero. In particolare, ad ogni ritrasmissione del frame dati viene calcolato e trasmesso un nuovo valore di T_{start}. Dal momento che non è stato mai ricevuto un frame di ACK in nessun tentativo di trasmissione, non è possibile sapere se il nodo N_t abbia effettivamente ricevuto un nuovo valore di T_{start}. Anche nel caso in cui ne avesse ricevuto uno, non sarebbe possibile risalire a quale valore tra tutti quelli trasmessi. Per questo motivo, il contatore `min_sleep_start` mantiene il più piccolo valore di T_{start} trasmesso, che è quello più restrittivo. Pertanto, se `min_sleep_start` è uguale a zero vuol dire che, nel caso in cui il ricevitore avesse ricevuto proprio il comando di sleep relativo al valore più piccolo di T_{start}, il link sul nodo N_t sarebbe già inattivo. Quindi, per evitare di perdere ulteriori frame in coda, anche N_i disattiva il link, che passa allo stato *closed*. Al contrario, un valore di `min_sleep_start` maggiore di zero indica che, se anche il ricevitore avesse ricevuto il valore di T_{start} minore, il link sul ricevitore sarebbe comunque ancora attivo e, di conseguenza, il link sul nodo trasmettitore rimane nello stato *planned*.

Anche lo stato *retr* ha subito delle piccole modifiche rispetto alla versione PRIL-MHB. In particolare, alla ricezione di un frame di ACK non si ha sempre la transizione verso lo stato *closed*. Infatti, quando viene trasmesso il comando di sleep con un valore di T_{start} maggiore di zero, alla ricezione del frame di ACK il link passa allo stato *planned*. Si ha la transizione verso lo stato *planned* anche quando, dopo aver effettuato N_{tries} tentativi, il frame dati viene scartato e il contatore `min_sleep_start` è maggiore di zero.

Invece, quando il frame dati viene scartato, ma il valore di `min_sleep_start` è pari a zero, il link viene disattivato e ha luogo la transizione verso lo stato *closed*. Come sempre, una vola che il link è nello stato *closed*, questo può ritornare allo stato *open* solo quando il contatore `sleep_end` raggiunge lo zero. Fino a quel momento, N_i non può trasmettere eventuali frame dati in coda.

Per quanto riguarda il calcolo del valore di T_{start}, la tecnica PRIL-MHI non impone nessuna specifica. La funzione può essere qualsiasi e può tenere conto di diversi parametri (ad esempio numero di ritrasmissioni, numero di frame dati nelle code dei figli, qualità del canale, ecc.). Tuttavia, è importante sottolineare che il valore di `sleep_start` deve essere sempre minore o al massimo uguale al valore di `sleep_end`. Di conseguenza, la funzione usata per il calcolo del valore di T_{start} non può restituire un valore maggiore di `sleep_end`. Per le simulazioni condotte

in questo lavoro di tesi è stata ideata e usata la seguente funzione:

$$T_{start} = Q_{virt} + \frac{N_{tries} - pkt.tries}{N_{tries}} Q_{real} + Q_{real}(queue\_size(target) - 1) \quad (4.1)$$

dove Q_{virt} e Q_{real} sono dei parametri statici definiti in fase di configurazione, N_{tries} è il massimo numero di ritrasmissioni per un singolo frame dati, $pkt.tries$ è il numero attuale di tentativi effettuati per il frame dati in trasmissione (il cui valore va da 0 fino a $N_{tries} - 1$) e $queue\_size(target)$ è il numero di pacchetti in coda da trasmettere al nodo target.

Impostando in fase di configurazione dei valori di Q_{virt} e Q_{real} pari a zero, la (4.1) restituisce sempre zero indipendentemente dal valore dagli altri parametri. Di conseguenza, dato che il link non può mai entrare nello stato *planned*, PRIL-MHI si comporta esattamente come PRIL-MHB. Questa caratteristica è voluta e non è da sottovalutare in quanto permette di passare in modo facile e veloce da una versione di PRIL-MH all'altra.

Un esempio di PRIL-MHI è illustrato in Figura 4.8. Nell'esempio è riportato lo stato di attivazione (sia dal lato del trasmettitore sia dal lato del ricevitore) di $N_f \rightarrow N_i$ e $N_i \rightarrow N_t$, che sono due link consecutivi che costituiscono parte del percorso seguito da tre pacchetti: P_1, P_2 e P_3. Il pacchetto P_1 è quello con periodo minimo (P_{min}) sia per il link $N_f \rightarrow N_i$ sia per il link $N_i \rightarrow N_t$. Durante lo slotframe 100, nel nodo N_f vengono accodati in ordine i pacchetti P_1 e P_2 e, nello stesso slotframe, N_f trasmette P_1 a N_i. Quest'ultimo aggiunge il comando di sleep e inoltra a sua volta il pacchetto a N_t. Nel comando di sleep viene indicato $T_{start} = 1$ e $T_{end} = 9$ (1/9). Quindi, il link $N_i \rightarrow N_t$ dal lato del nodo N_i passa allo stato *planned*, mentre il nodo N_t, seguendo le informazioni ricevute, inizializza il suo contatore interno `sleep_start` con il valore 1 e mantiene il link nello stato *open*. Pertanto, quando durante lo slotframe 101 il nodo N_i riceve anche il pacchetto P_2, questo non rimane bloccato in coda, ma può essere trasmesso subito a N_t. Durante la trasmissione di P_2, il nodo N_i posticipa ulteriormente la disattivazione del link, trasmettendo il comando di sleep con $T_{start} = 1$. Di conseguenza, nello slotframe 102, il link $N_i \rightarrow N_t$ rimane ancora attivo. In questo caso, però, non essendoci trasmissioni da parte di N_i, si verifica il fenomeno di idle listening sul nodo N_t. Allo stesso modo, si ha idle listening anche sul nodo N_i in quanto il link è mantenuto attivo nello slotframe 102, ma il nodo N_f non ha frame dati in coda da trasmettere. Dallo slotframe 103 allo slotframe 109 entrambi i link restano nello stato *closed*. Durante lo slotframe 104, nel nodo N_f viene accodato il pacchetto P_3. Il link $N_f \rightarrow N_i$, però, è ancora inattivo e tale pacchetto rimane bloccato in coda fino allo slotframe 110, ossia fino alla prossima riattivazione del link.

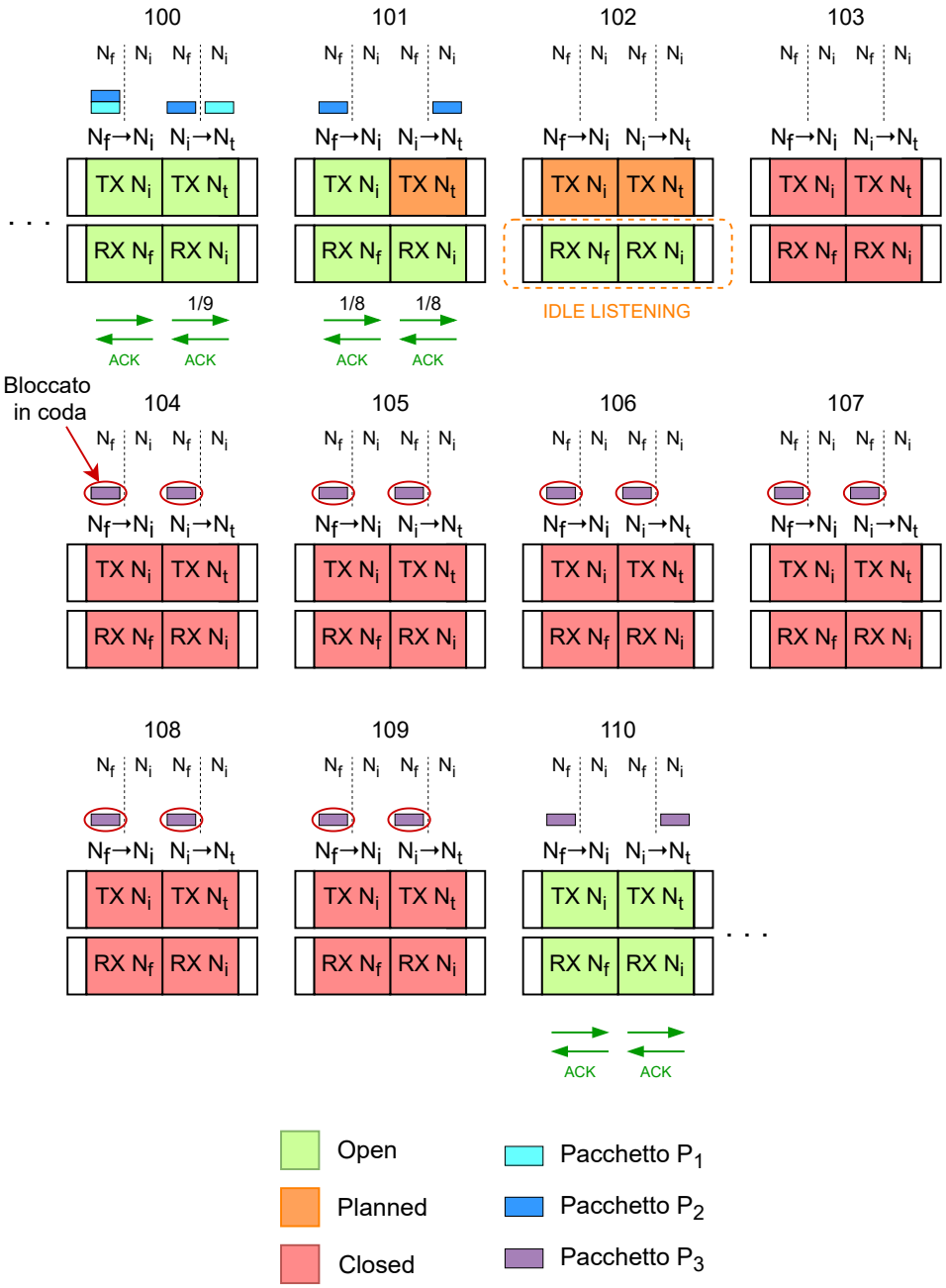


Figura 4.8: Esempio sul funzionamento della tecnica PRIL-MHI

Questo esempio mostra come, ritardando la disattivazione del link, è possibile evitare che i pacchetti già presenti in coda rimangano bloccati nel hop successivo, come accade in PRIL-MHB. Tuttavia, PRIL-MHI non è in grado di fare nulla per i pacchetti accodati quando il link verso il prossimo hop è già nello stato *closed*. Infatti, questi pacchetti rimangono bloccati nella coda fino a quando il link non

ritorna attivo, il che provoca un aumento delle latenze di comunicazione. La soluzione ideata per risolvere questo problema è quella di fare in modo che il link, una volta disattivato, si riattivi ad intervalli regolari per dare modo al trasmettitore di trasmettere eventuali frame dati in coda, senza dover necessariamente aspettare il ritorno del link allo stato *open*. Questo ha portato allo sviluppo di una nuova versione di PRIL-MH, chiamata PRIL-MHI2, che estende il funzionamento di PRIL-MHI per introdurre un meccanismo di riattivazione ciclica del link.

4.5 PRIL-MHI2

Come spiegato in precedenza, una delle grosse limitazioni di PRIL-MHB e PRIL-MHI è quella di aumentare di molto le latenze di comunicazione. Questo risulta essere particolarmente vero nei casi in cui i pacchetti vengono generati in momenti diversi. In contesti applicativi reali, questa situazione è molto comune in quanto le applicazioni in esecuzione sui nodi sorgente non sono sincronizzate tra loro nella generazione dei pacchetti. Questo causa un sostanziale aumento delle latenze che nemmeno PRIL-MHI, posticipando la disattivazione del link, può evitare.

Per limitare l'aumento delle latenze, pur mantenendo bassi i consumi energetici dei nodi della rete, è stata sviluppata una nuova versione della tecnica PRIL-MHI, chiamata PRIL-MHI2, basata sulla riattivazione ciclica del link. In fase di configurazione viene deciso un periodo di riattivazione del link che sia adeguato per soddisfare i requisiti dell'applicazione in termini di latenze di comunicazione. Quindi, una volta che il link è stato disattivato, trasmettitore e ricevitore lo riattivano periodicamente in modo tale che eventuali frame dati in coda possano avanzare verso il prossimo hop in tempi più brevi. Questo meccanismo, una volta attivato, ha effetto fino a quando il link è nello stato *closed* e la sua validità termina al ritorno allo stato *open*.

Rispetto alla versione PRIL-MHI, al comando di sleep da usare in PRIL-MHI2 è stato aggiunto un nuovo campo, T_{react}, che indica al nodo target ogni quante ripetizioni della cella deve riattivare il link. Tuttavia, ci sono situazioni in cui nel comando di sleep scambiato tra due nodi che operano usando PRIL-MHI2 non è presente il campo T_{react}. Infatti, il meccanismo di riattivazione ciclica non viene attivato per tutti i link della rete, ma solo in quelli per cui il periodo minimo di riferimento (P_{min}) è maggiore del valore del periodo di riattivazione scelto in fase di configurazione. Nei link per cui il valore di P_{min} è minore, il meccanismo di riattivazione periodica è inutile in quanto il link ritorna sempre allo stato *open* prima del tempo indicato in T_{react}. In questo caso, PRIL-MHI2 si comporta allo stesso modo di PRIL-MHI.

Per poter funzionare in modo opportuno, il meccanismo di riattivazione ciclica richiede che il nodo N_i e il nodo N_t siano sincronizzati sul tempo in cui entrambi

devono riattivare il link. Se così non fosse, N_i proverebbe a effettuare delle trasmissioni quando il link su N_t è inattivo e, allo stesso modo, il nodo N_t ascolterebbe il canale quando il link sul nodo N_i è inattivo. Il protocollo di sincronizzazione definito in PRIL-MHI2 prevede che, ad ogni trasmissione (ritrasmissioni incluse), il trasmettitore includa sempre nel comando di sleep il campo T_{react} e che, ad ogni ricezione del comando di sleep contenente il campo T_{react}, il ricevitore reinizializzi il suo contatore interno `activation_countdown`. La sincronizzazione viene portata a termine con successo solo alla ricezione del frame di ACK e il trasmettitore non deve più includere il campo T_{react} nel comando di sleep. In questo modo il trasmettitore ha la certezza di riattivare il link nelle stesse celle in cui lo farà anche il ricevitore. Al contrario, se non viene mai ricevuto un frame di ACK relativo ad un frame dati contenente il campo T_{react}, la sincronizzazione non va a buon fine. In questo caso il trasmettitore non può sapere se il ricevitore riattiverà il link periodicamente e in quali celle. Pertanto, il trasmettitore non può abilitare il meccanismo per l'intero periodo di disattivazione del link. Tuttavia, il ricevitore potrebbe aver ricevuto il comando di sleep con il campo T_{react} e, quindi, potrebbe aver abilitato la riattivazione ciclica. In questo caso, in ogni cella in cui il ricevitore attiva il link, si ha il fenomeno di idle listening.

Ad ogni modo, indipendentemente dal fatto che la sincronizzazione abbia avuto successo o meno, quando il link ritorna allo stato *open* è necessario effettuare nuovamente il processo di sincronizzazione per poter utilizzare il meccanismo di riattivazione periodica nel periodo successivo.

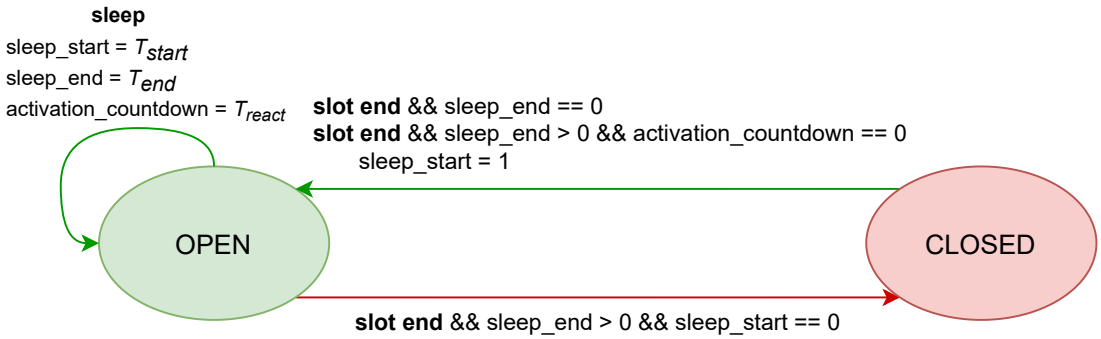


Figura 4.9: Macchina a stati del nodo ricevitore in PRIL-MHI2

Anche in questo caso, per descriverne il funzionamento, l'algoritmo della tecnica PRIL-MHI2 è stato schematizzato usando una macchina a stati. In Figura 4.9 è stata riportata la macchina a stati specifica per il ricevitore (N_t). Dal momento che il funzionamento ha subito piccole variazioni rispetto alla tecnica PRIL-MHI, verranno illustrate solo le novità introdotte.

Ogni volta che N_t riceve il comando di sleep contenente il campo T_{react}, ne estrae il valore e lo usa per inizializzare il suo contatore interno `activation_countdown`. Quest'ultimo viene decrementato ad ogni ripetizione della cella $N_i \rightarrow N_t$. Quando il link è nello stato *closed*, ogni volta che `activation_countdown` arriva a zero, si ha la transizione verso lo stato *open*, e il contatore viene reinizializzato in modo che ricominci da subito a tenere traccia della prossima riattivazione. L'attivazione del link, però, non deve essere definitiva. Quindi, prima di passare allo stato *open*, il contatore `sleep_start` viene inizializzato con il valore 1 per indicare che il link può rimanere attivo solo per una cella. Infatti, quando il `sleep_start` arriva a zero, il link ritorna di nuovo nello stato *closed*.

In Figura 4.10 è riportata la macchina a stati del trasmettitore (N_i). Anche in questo caso le modifiche apportate sono minime. Il meccanismo di riattivazione periodica prevede che il link ritorni attivo solo per il tempo necessario a trasmettere eventuali frame dati in coda, per poi tornare allo stato *closed*. Questo comportamento è lo stesso che si ha quando il link è nello stato *planned*. Infatti, in questo stato, c'è già un periodo di disattivazione impostato, ma, prima di passare allo stato *closed*, il link rimane temporaneamente attivo per il periodo di tempo indicato in `sleep_start`. Quindi, sfruttando lo stato *planned* già implementato in PRIL-MHI, è possibile integrare in modo facile il meccanismo di riattivazione periodica.

In particolare, quando riesce a sincronizzarsi con il ricevitore, il trasmettitore inizializza il suo contatore interno `activation_countdown` a partire dall'ultimo valore di T_{react} trasmesso. Il valore del contatore viene decrementato ad ogni ripetizione della cella $N_i \rightarrow N_t$. Ogni volta che `activation_countdown` raggiunge lo zero mentre il link è nello stato *closed*, il contatore `sleep_start` viene inizializzato con il valore 1 e ha luogo la transizione verso lo stato *planned*. Inoltre, in questa occasione viene anche ripristinato il valore di `activation_countdown` in modo sia già impostato sul tempo in cui si avrà la prossima riattivazione. Dal momento che `sleep_start` viene inizializzato con il valore 1, il link rimarrà attivo solo per una cella prima di tornare nello stato *closed*. Tuttavia, come spiegato in Sezione 4.4, ad ogni ritrasmissione può essere calcolato e trasmesso un nuovo valore di T_{start} ed è, quindi, possibile posticipare la disattivazione del link della quantità di tempo necessaria per trasmettere ulteriori frame dati in coda. Quando il contatore `sleep_start` arriva a zero, il link ritorna nello stato *closed*. Quando il contatore `activation_countdown` raggiunge nuovamente lo zero, l'intero processo descritto finora si ripete.

Invece, quando è il contatore `sleep_end` ad arrivare a zero, il link passa allo stato *open*. Quindi, il meccanismo di riattivazione periodica viene disabilitato e il trasmettitore dovrà sincronizzarsi di nuovo con il ricevitore per abilitarlo anche nel periodo successivo.

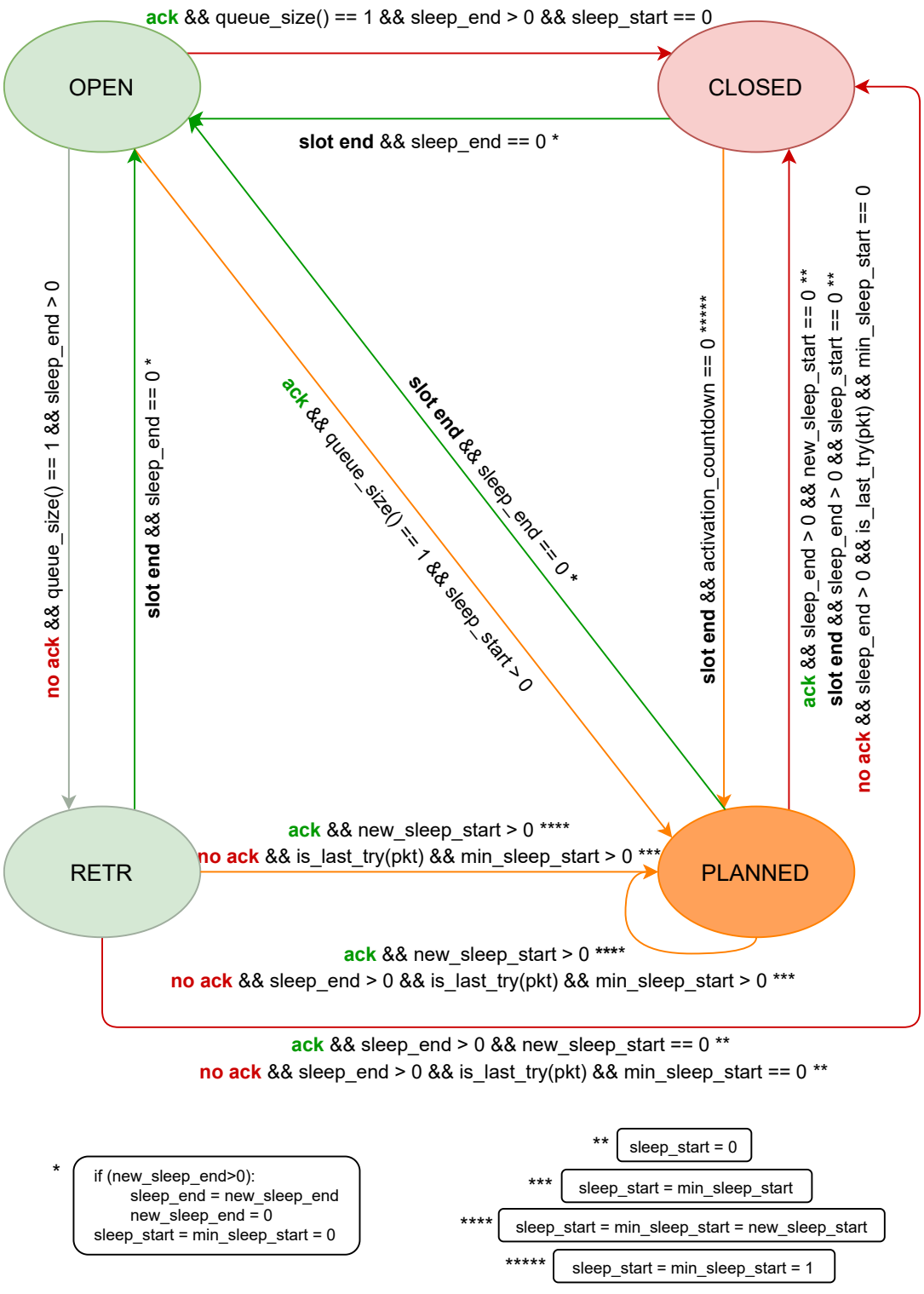


Figura 4.10: Macchina a stati del nodo trasmettitore in PRIL-MHI2

In Figura 4.11 è riportato un esempio che illustra il funzionamento della tecnica PRIL-MHI2. In particolare, vengono seguite le trasmissioni di tre pacchetti (P_1, P_2 e P_3) sui link $N_f \rightarrow N_i$ e $N_i \rightarrow N_t$, i quali fanno parte del percorso seguito dai pacchetti per arrivare a destinazione. Il pacchetto P_1 appartiene al flusso con periodo di generazione minimo ed è usato come riferimento sia per il link $N_f \rightarrow N_i$ sia per il link $N_i \rightarrow N_t$. Per ogni cella è mostrato sia lo stato di attivazione del link dal lato del trasmettitore sia lo stato di attivazione del link dal lato del ricevitore.

Durante lo slotframe 100, nel nodo N_f vengono accodati in ordine i pacchetti P_1 e P_2. Nello stesso slotframe, N_f trasmette il pacchetto P_1 al nodo N_i. Quest'ultimo aggiunge il comando di sleep a P_1 e inoltra il pacchetto a N_t. Nel comando di sleep, oltre a $T_{start} = 1$ e $T_{end} = 9$, viene incluso anche $T_{react} = 3$, che indica al ricevitore di riattivare il link ogni tre ripetizioni della cella. Dal momento che viene ricevuto il frame di ACK, la sincronizzazione tra N_i e N_t ha successo ed è possibile sfruttare il meccanismo di riattivazione periodica sul link $N_i \rightarrow N_t$. In modo analogo, N_f e N_i si sincronizzano durante la trasmissione del pacchetto P_2 nello slotframe 101. Pertanto è possibile usare la riattivazione periodica anche sul link $N_f \rightarrow N_i$. Si fa notare che, durante la trasmissione del pacchetto P_2 nello slotframe 101, il nodo N_i non aggiunge al comando di sleep il campo T_{react} in quanto la sincronizzazione con il nodo N_t ha già avuto successo.

A partire dallo slotframe 103, entrambi i link passano allo stato *closed*. Da questo momento il meccanismo di riattivazione periodica entra in azione. Infatti, il link $N_f \rightarrow N_i$ si riattiva negli slotframe 104 e 107, mentre il link $N_i \rightarrow N_t$ si riattiva durante gli slotframe 106 e 109. Grazie a questo meccanismo, quando durante lo slotframe 104 viene accodato nel nodo N_f il pacchetto P_3, questo può essere trasmesso subito al nodo N_i. Inoltre, la riattivazione del link $N_i \rightarrow N_t$ permette al nodo N_i di inoltrare P_3 a N_t già durante lo slotframe 106, senza aspettare che il link torni allo stato *open* nello slotframe 110.

Da questo esempio è possibile vedere come PRIL-MHI2 permetta di limitare l'aumento delle latenze di comunicazione per i pacchetti accodati quando il link verso il prossimo hop non è più attivo. Tuttavia, questa soluzione ha lo svantaggio di aumentare leggermente i consumi energetici. Infatti, quando il trasmettitore non ha frame dati in coda, tutte le volte che il ricevitore riattiva il link per ascoltare il canale si ha il fenomeno di idle listening.

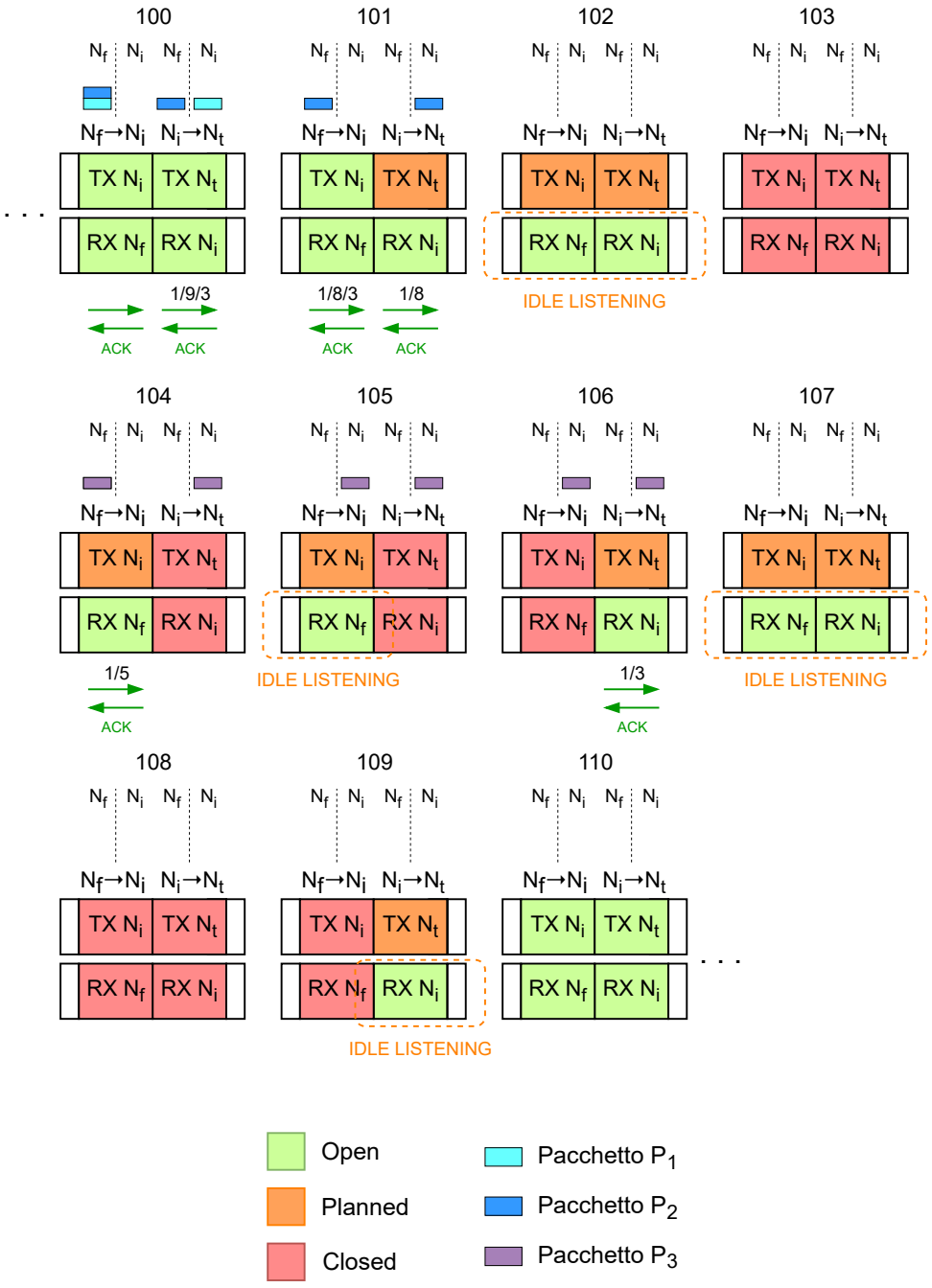


Figura 4.11: Esempio sul funzionamento della tecnica PRIL-MHI2

Capitolo 5

Implementazione

5.1 TSCH-predictor

Per le simulazioni effettuate in questa tesi è stato usato il simulatore ad eventi discreti `TSCH-predictor`, già implementato in [3] per valutare l'efficacia della tecnica PRIL-F e successivamente migliorato in [20]. Il simulatore è stato progettato per poter analizzare in molto semplice il comportamento delle reti TSCH, focalizzandosi solo sulle parti del protocollo strettamente essenziali per ottenere degli indici prestazionali di interesse. Rispetto ad altri simulatori a eventi discreti quali 6TiSCH simulator [21] e TSCH-Sim [22], `TSCH-predictor` permette di implementare e valutare nuovi algoritmi in modo semplice e veloce.

Ovviamente, avendo implementato solo le parti rilevanti del protocollo TSCH, il simulatore presenta alcune limitazioni. Ad esempio, lo matrice TSCH che definisce lo schedule globale della rete è configurabile solo staticamente e non prevede la possibilità di avere celle di tipo shared. Anche le topologia di rete è statica e, pertanto, i nodi non possono essere spostati durante la simulazione. Inoltre, non sono modellate le trasmissioni dei messaggi di controllo (ad esempio quelli generati dal protocollo RPL), ma solo quelle relative al traffico generato dalle applicazioni.

Tuttavia, il simulatore permette una facile configurazione dei parametri principali della modalità operativa TSCH. Infatti, è possibile dettagliare la struttura dello slotframe, indicando il numero di timeslot che lo compongono (N_{slot}) e la durata del singolo timeslot (T_{slot}). Inoltre, è possibile specificare il numero massimo di ritrasmissioni consentite (N_{tries}) per un singolo frame di dati, prima che questo venga scartato.

Anche la procedura da seguire per la configurazione della matrice TSCH è molto semplice. Infatti, per ogni cella che si vuole allocare è sufficiente specificare lo *slotOffset*, il *channelOffset*, l'identificativo del nodo trasmettitore e l'identificativo del nodo ricevitore. Il simulatore supporta anche l'overprovisioning e, pertanto, consente di allocare alla stessa coppia di nodi più celle della matrice TSCH per formare un bundle. Tuttavia, dal momento che lo schedule è statico e non può

essere modificato durante la simulazione, non è possibile aggiungere o rimuovere dinamicamente celle al bundle sulla base del traffico attuale. Inoltre, per ogni canale è possibile specificare la probabilità di errore separatamente per i frame dati (ϵ_{data}) e per i frame di ACK (ϵ_{ack}).

Una volta definito lo schedule globale, è necessario specificare quali sono i flussi dati presenti nella rete in modo tale che venga simulata la generazione di nuovi pacchetti secondo le indicazioni fornite. La versione attuale di `TSCH-predictor` supporta solo il traffico di tipo periodico e, quindi, per ogni flusso che si vuole definire è necessario indicare, oltre all'identificativo del nodo sorgente e quello del nodo destinazione, anche il periodo di generazione dell'applicazione. Dal momento che non è stato previsto il supporto per i protocolli di routing, per ogni flusso bisogna specificare staticamente il percorso che i pacchetti devono seguire per raggiungere la destinazione.

Infine, il simulatore permette di scegliere il modello energetico da usare per il calcolo delle statistiche sui consumi. In particolare, bisogna indicare l'energia necessaria per trasmettere un frame dati (E_{tx}), l'energia necessaria per ricevere un frame dati (E_{rx}) e l'energia che un nodo consuma in idle listening (E_{idle}). Queste quantità devono essere espresse in joule e si riferiscono ai consumi relativi a un singolo frame dati di dimensione massima (127 byte) [23].

Il simulatore `TSCH-predictor` è stato completamente implementato usando il linguaggio di programmazione `python` ed è basato su `SimPy`, che è un framework di simulazione ad eventi discreti. Il software è strutturato in diverse classi: `TSCHpredictor`, `TSCH`, `Node`, `Packet`, `Generator` e `Logger`. La classe `TSCHpredictor` è il punto di ingresso del programma e ha il compito di leggere i file di configurazione per inizializzare tutte le strutture dati necessarie alla simulazione. Innanzitutto, crea una rappresentazione interna della matrice TSCH e dello slotframe. Successivamente, configura opportunamente il modulo `SimPy`, impostando la durata della simulazione e creando gli eventi da simulare. In particolare, per ogni cella dello schedule viene creato un evento periodico che `SimPy` esegue ciclicamente durante il relativo timeslot dello slotframe. Tali eventi non sono direttamente gestiti nella classe `TSCHpredictor`, ma sono affidati al modulo `TSCH`. Quest'ultimo gestisce le comunicazioni tra i nodi della rete per fare in modo che queste avvengano secondo le modalità stabilite dal protocollo TSCH dello standard IEEE 802.15.4.

La generazione dei pacchetti è affidata al modulo `Generator` che, a partire dai flussi definiti in fase di configurazione, crea degli eventi in `SimPy` per poter simulare la generazione di nuovi pacchetti da parte delle applicazioni. In particolare, seguendo le indicazioni fornite per ogni flusso, vengono creati periodicamente degli oggetti di tipo `Packet` da aggiungere alla coda del nodo sorgente. La classe `Packet` fornisce una rappresentazione basilare dei pacchetti scambiati tra i nodi di una rete TSCH. Nel simulatore ogni pacchetto è univocamente identificato da un `id` e contiene varie informazioni quali l'identificativo del nodo sorgete, l'identificativo del

nodo destinazione e il percorso da seguire per raggiungere la destinazione, nonché una serie di informazioni statistiche relative al pacchetto quali la latenza e il numero di ritrasmissioni. Inoltre, la classe `Packet` permette di manipolare facilmente il campo information element del frame dati attraverso la variabile di istanza `ie`.

La classe `Node` rappresenta un nodo di una rete TSCH. Ogni nodo è univocamente identificato da un `id` e mantiene, per ognuno dei nodi confinanti, una coda dei pacchetti da trasmettere. Inoltre, i nodi tengono traccia del numero di celle effettivamente usate per trasmettere un frame dati, del numero di celle in cui è stato ricevuto un frame dati e del numero di celle passate in idle listening. Sulla base di questi dati, il simulatore è in grado di mostrare delle statistiche sui consumi energetici di ogni singolo nodo.

Infine, il modulo `Logger` è utilizzato per tenere traccia, salvandole su file, di varie informazioni legate ai pacchetti quali il tempo di generazione, il tempo di arrivo e il numero di ritrasmissioni. In questo modo è possibile valutare la latenza e l'affidabilità della rete.

5.2 Software realizzato

Per poter verificare l'efficacia delle tecniche PRIL-MH e valutarne le prestazioni, queste sono state implementate nel simulatore `TSCH-predictor`.

Innanzitutto, sono stati aggiunti dei nuovi parametri di configurazione, che adesso includono anche Q_{virt}, Q_{real} e $T_{riattivazione}$. I primi due sono usati in (4.1) per il calcolo del valore di T_{start}, mentre $T_{riattivazione}$ indica il periodo da usare per il meccanismo di riattivazione ciclica implementato in PRIL-MHI2. Il software realizzato è unico per tutte e tre le versioni di PRIL-MH ed è possibile ottenere il comportamento di una specifica tecnica impostando opportunamente i parametri di configurazione appena presentati. In particolare, per ottenere il comportamento base della tecnica (PRIL-MHB) è necessario usare il valore zero per tutti e tre i parametri. Scegliendo un valore maggiore di zero per almeno uno tra Q_{virt} e Q_{real} si passa alla versione PRIL-MHI. Infine, per ottenere il meccanismo di riattivazione ciclica offerto da PRIL-MHI2 è necessario configurare il parametro $T_{riattivazione}$ con un valore maggiore di zero.

Per quanto riguarda il codice, le principali modifiche sono state apportate nei moduli `TSCH` e `Node`. Dal momento che uno stesso nodo può agire sia come trasmettitore sia come ricevitore e che, a seconda del caso, l'algoritmo ha bisogno di variabili diverse per gestire lo stato di attivazione del link, nella classe `Node` sono state introdotte due strutture dati separate: `targets_status` e `children_status`. La variabile `targets_status` è una collezione di oggetti di tipo `dictionary`, ognuno dei quali contiene le informazioni per gestire lo stato di attivazione del link quando il nodo agisce da trasmettitore. In questo caso, per ogni nodo target, bisogna memorizzare il flusso con periodo minimo di riferimento per il link

(P_{min} e N_{ref}), il flusso di riserva (P_{backup} e N_{backup}) e il valore di tutti i contatori usati dal trasmettitore (`sleep_start`, `new_sleep_start`, `min_sleep_start`, `sleep_end`, `new_sleep_start` e `activation_countdown`).

La variabile `children_status` rappresenta una collezione di oggetti di tipo `dictionary` ed è usata per gestire lo stato di attivazione dei link in cui il nodo agisce da ricevitore. In questo caso, per ogni nodo figlio da cui riceve il comando di sleep, il ricevitore deve memorizzare i valori dei contatori `sleep_start`, `sleep_end` e `activation_countdown`. Per entrambe le strutture dati, l'accesso alle informazioni di uno specifico link avviene attraverso l'identificativo dell'altro nodo incluso nelle comunicazioni, che è il nodo target per `targets_status` e il nodo figlio per `children_status`.

L'implementazione della tecnica PRIL-MH ha richiesto la modifica dell'interfaccia pubblica della classe `Node`, nella quale sono stati integrati i seguenti metodi: `can_transmit()`, `can_receive()`, `on_transmission()`, `on_reception()`, `on_ack()`, `on_ack_loss()` e `on_slot_end()`. Tali metodi, opportunamente invocati dal modulo `TSCH`, gestiscono tutte le fasi dell'algoritmo sia per il nodo trasmettitore che per il nodo ricevitore. Per mostrare l'ordine e il contesto in cui questi sono invocati, di seguito è riportata una versione semplificata del metodo `tsch_event()` della classe `TSCH`.

```
def tsch_event(cell: Cell):
  transmitter.on_transmission(cell)
  pkt = transmitter.next_pkt_in_queue(receiver.id)
  # DATA frame transmission
  if pkt is not None and transmitter.can_transmit(cell):
    if random.uniform(0,1) > cell.frame_error_prob
      and receiver.can_receive(cell):
        receiver.on_reception(cell, pkt)
        # ACK frame transmission
        if random.uniform(0,1) > cell.ack_error_prob:
          transmitter.on_ack(cell, pkt)
        else:
          transmitter.on_ack_loss(cell, pkt)
    else:
      receiver.on_reception(cell, None)
      transmitter.on_ack_loss(cell, pkt)
  else:
    receiver.on_reception(cell, None)
  transmitter.on_slot_end(cell)
```

Listing 5.1: Metodo `tsch_event()` del modulo `TSCH`

Prima di trasmettere un pacchetto viene sempre verificato, attraverso il metodo `can_transmit()`, lo stato di attivazione del link sul nodo trasmettitore. Solo se il link risulta attivo si procede alla trasmissione e si verifica che il ricevitore stia effettivamente ascoltando il canale (`can_receive()`). Tuttavia, quest'ultima condizione non è sufficiente a garantire che il ricevitore riesca a ricevere il pacchetto.

Infatti, è necessario verificare che questo non subisca errori durante la trasmissione. L'errore sul canale è stato implementato nel simulatore attraverso la generazione di un numero casuale nell'intervallo $[0, 1]$, il quale viene confrontato con il valore di errore scelto in fase di configurazione (ϵ_{data}) per ottenere la probabilità di successo voluta. Se il pacchetto arriva integro al ricevitore, si procede con la trasmissione del frame di ACK. Anche in questo caso è necessario verificare che il frame arrivi correttamente al trasmettitore e, a seconda della situazione, viene eseguito il metodo `on_ack()` o `on_ack_loss()`. Il metodo `on_slot_end()` è l'ultimo ad essere invocato ed è usato per notificare al trasmettitore la fine del timeslot.

Si fa notare che i metodi `on_transmission()` e `on_reception()` sono eseguiti indipendentemente dallo stato di attivazione del link. Infatti, questi gestiscono anche il decremento dei contatori necessari per il funzionamento di PRIL-MH e, pertanto, devono essere invocati sempre in ogni cella della matrice TSCH riservata alle comunicazioni. Solo nel caso in cui il pacchetto riesce effettivamente ad arrivare al nodo ricevitore, il metodo `on_reception()` riceve il riferimento del pacchetto trasmesso. In tutti gli altri casi viene passato il valore `None` per segnalare al nodo la mancata ricezione.

Passando ora alla classe `Node`, di seguito verranno forniti maggiori dettagli sui nuovi metodi aggiunti, riportando e spiegando per ognuno le parti rilevanti del codice realizzato.

```
def on_transmission(self, cell: Cell) -> None:
  target_id = cell.receiver_id
  target = self.targets_status[target_id]

  self._update_counters_transmitter(target)
  if self.queue_size(target_id) > 0:
    if target['state'] == State.open: self._open(target_id)
    if target['state'] == State.retr: self._retr(target_id)
    if target['state'] == State.planned: self._planned(target_id)
    # Otherwise closed
```

Listing 5.2: Metodo `on_transmission()` del modulo `Node`

Il metodo `on_transmission()`, la cui implementazione è riporta sopra, ha due compiti fondamentali, ossia decrementare i valori dei contatori usati dal trasmettitore sul link con il `target` ed, eventualmente, aggiungere il comando di sleep ai pacchetti in trasmissione. In particolare, il decremento dei contatori avviene attraverso la funzione `_update_counters_transmitter()`, mentre il codice relativo alla manipolazione del pacchetto in trasmissione è stato diviso in tre metodi (`_open()`, `_retr()` e `_planned()`) sulla base dello stato del link. Il codice implementato in ogni funzione segue fedelmente i passaggi descritti durante la spiegazione della macchina a stati nel Capitolo 4 e, per questo motivo, di seguito è riportata solo l'implementazione per lo stato *planned*.

```
def _planned(self, target_id) -> None:
  target = self.targets_status[target_id]
  pkt = self.next_pkt_in_queue(target_id)
  queue_size = self.queue_size(target_id)

  target['new_sleep_start'] = self._get_sleep_start(target_id)
  target['min_sleep_start'] = min(target['new_sleep_start'],
                                  target['min_sleep_start'])

  pkt.ie['start'] = target['new_sleep_start']
  pkt.ie['end'] = target['sleep_end']
  pkt.ie['queue_size'] = queue_size - 1

  if target['react_value'] > 0:
    pkt.ie['react'] = target['react_value']
```

Listing 5.3: Metodo `_planned()` del modulo `Node`

In `_planned()`, per modificare l'inizio del periodo di disattivazione del link, il contatore `new_sleep_start` viene ricalcolato ad ogni trasmissione. Per il calcolo è usata la funzione `_get_sleep_start()`, che rappresenta l'implementazione in `python` di (4.1). Successivamente si procede con l'aggiunta del comando di sleep nell'information element del pacchetto. Il campo `react`, che indica il periodo della riattivazione ciclica, non è sempre aggiunto all'information element, ma la decisione è presa sulla base del valore di `react_value`. Infatti, come spiegato in Sezione 4.5, il comando di sleep deve includere il campo T_{react} solo quando si intende abilitare il meccanismo di riattivazione ciclica e solo per il tempo necessario affinché il trasmettitore e il ricevitore si sincronizzino. Quindi, ogni volta che è necessario attivare tale meccanismo, `react_value` viene inizializzato con il valore da trasmettere nel campo `react`.

```
def on_ack(self, cell: Cell, pkt: Packet) -> None:
  target_id = cell.receiver_id
  target = self.targets_status[target_id]
  [...]
  if target['state'] == State.planned:
    if target['new_sleep_start'] > 0:
      target['sleep_start'] = target['new_sleep_start']
      target['min_sleep_start'] = target['new_sleep_start']
    elif target['sleep_end'] > 0
      and target['new_sleep_start'] == 0:
        target['state'] = State.closed
        target['sleep_start'] = 0
  [...]
  if 'react' in pkt.ie:
    target['activation_period'] = pkt.ie['react']
    target['activation_countdown'] = pkt.ie['react'] - 1
    target['react_value'] = 0
```

Listing 5.4: Metodo `on_ack()` del modulo `Node`

Per rendere maggiormente chiara l'utilità di `react_value`, è stata riportata la parte di codice del metodo `on_ack()` in cui viene gestita la conclusione del processo di sincronizzazione. In questo caso il campo `react` non deve più essere incluso nel comando di sleep e, quindi, `react_value` viene riportato a zero. In questa occasione vengono anche inizializzate le variabili `activation_countdown` e `activation_period`. Quest'ultima mantiene il periodo della riattivazione ciclica ed è utilizzata per reinizializzare il contatore `activation_countdown` ogni volta che questo raggiunge lo zero. Nel metodo `on_ack()` sono anche gestite tutte le transizioni innescate dalla ricezione del frame di ACK. A titolo esemplificativo, è stata riportata la parte di codice relativa alle transizioni effettuate quando il nodo è nello stato *planned*.

```
def on_ack_loss(self, cell: Cell, pkt: Packet) -> None:
  target_id = cell.receiver_id
  target = self.targets_status[target_id]
  [...]
  if target['state'] == State.planned:
    if target['sleep_end'] > 0
      and pkt.is_last_try(self.id)
      and target['min_sleep_start'] > 0:
        target['sleep_start'] = target['min_sleep_start']
    elif target['sleep_end'] > 0
      and pkt.is_last_try(self.id)
      and target['min_sleep_start'] == 0:
        target['state'] = State.closed
        target['sleep_start'] = 0
  [...]
```

Listing 5.5: Metodo `on_ack_loss()` del modulo `Node`

Il metodo `on_ack_loss()` si occupa esclusivamente della gestione dei passaggi di stato da effettuare a seguito della perdita del frame di ACK. Anche in questo caso è stato riportato solo il frammento di codice che gestisce le transizioni quando il link è nello stato *planned*. La funzione `is_last_try()` è usata per verificare che quello appena effettuato sia l'ultimo degli N_{tries} tentativi che il nodo ha avuto a disposizione per trasmettere il pacchetto.

L'ultima funzione invocata sul nodo trasmettitore è `on_slot_end()`, che contiene il codice da eseguire alla fine della cella. In questo metodo vengono gestite tutte le transizioni legate all'arrivo dei contatori a zero. Le transizioni più interessanti da analizzare sono quelle relative alla riattivazione del link quando questo si trova nello stato *closed*. Pertanto, di seguito è stato riportato il codice eseguito dal nodo trasmettitore per gestire il ritorno allo stato *open* e il passaggio temporaneo allo stato *planned*.

```
def on_slot_end(self, cell: Cell) -> None:
  target_id = cell.receiver_id
  target = self.targets_status[cell.receiver_id]
  [...]
  if target['state'] == State.closed:
    if target['sleep_end'] == 0:
      target['activation_period'] = 0
      target['activation_countdown'] = 0
      target['react_value'] = 0
      if target['new_sleep_end'] > 0:
        target['sleep_end'] = target['new_sleep_end']
        target['new_sleep_end'] = 0
        if CONF_REACT_PERIOD < target['min_period']:
          target['react_value'] = n_cell_period(self.id, target_id,
                                                CONF_REACT_PERIOD)
      target['sleep_start'] = target['min_sleep_start'] = 0
      target['state'] = State.open
  elif target['sleep_end'] > 0
    and target['activation_period'] > 0
    and target['activation_countdown'] == 0:
      target['sleep_start'] = target['min_sleep_start'] = 1
      target['activation_countdown'] = target['activation_period']
      target['state'] = State.planned
```

Listing 5.6: Metodo `on_slot_end()` del modulo `Node`

Quando il contatore `sleep_end` raggiunge lo zero, il link deve passare allo stato *open*. Prima di farlo vengono azzerati tutti i campi relativi alla riattivazione ciclica (`activation_period`, `activation_countdown` e `react_value`). Successivamente si controlla se è già stato ricevuto il prossimo pacchetto con periodo di generazione minimo (`new_sleep_end > 0`) e, in caso affermativo, il contatore `sleep_end` viene subito reinizializzato prima di passare allo stato *open*. Nel caso in cui il periodo scelto per la riattivazione ciclica (`CONF_REACT_PERIOD`) sia minore del periodo minimo di riferimento, è necessario attivare tale meccanismo per il link. Quindi, `react_value` deve essere inizializzato per poter innescare il processo di sincronizzazione. Per il calcolo viene usata la funzione `n_cell_period()` che analizza la matrice TSCH e restituisce il numero di volte che le celle appartenenti al link tra questo nodo e il `target` si ripeteranno nel prossimo periodo temporale di lunghezza pari a `CONF_REACT_PERIOD`. A questo punto il link ritorna allo stato *open*.

Invece, se è il contatore `activation_countdown` ad aver raggiunto lo zero, il link deve riattivarsi temporaneamente. Quindi, `sleep_start` viene impostato con il valore 1 e viene esplicitato il passaggio allo stato *planned*. In questa occasione viene anche ripristinato il valore del contatore `activation_countdown` in modo tale che parta subito il conteggio per la riattivazione successiva.

Per quanto riguarda il ricevitore, il codice è interamente contenuto nel metodo `on_reception()` e può essere logicamente diviso in due parti: la prima in cui viene

gestito l'arrivo del comando di sleep e la seconda in cui viene gestito l'arrivo del pacchetto con periodo minimo.

```
def on_reception(self, cell: Cell, pkt: Packet) -> None:
  child = self.children_status[cell.transmitter_id]
  self._update_counters_receiver(child)
  if pkt is None: return
  if 'start' in pkt.ie and 'end' in pkt.ie:
    child['sleep_start'] = pkt.ie['start']
    child['sleep_end'] = pkt.ie['end']
    if 'react' in pkt.ie:
      child['activation_period'] = pkt.ie['react']
      child['activation_countdown'] = pkt.ie['react'] - 1
  [...]
```

Listing 5.7: Metodo `on_reception()` del modulo `Node`: gestione del comando di sleep

In `on_reception()`, dopo aver decrementato i contatori usati dal ricevitore sul link usando `_update_counters_receiver()`, si controlla che il pacchetto contenga il comando di sleep. In questo caso i valori dei campi contenuti nell'information element vengono usati per reimpostare i contatori interni. Solo se è presente anche il campo `react`, il ricevitore deve conservarne il valore in `activation_period` e inizializzare il suo contatore `activation_countdown`.

```
  [...]
  if self.is_dup_packet(pkt) == False:
    src_id = pkt.get_source_node()
    period = pkt.ie['period']
    target_id = pkt.get_next_hop(self.id)
    target = self.targets_status[target_id]

    if target['min_period']==period and target['ref_node']==src_id:
      if target['sleep_end'] == 0:
        target['sleep_end'] = n_cell_period(self.id, target_id,
                                            period)
        if CONF_REACT_PERIOD < target['min_period']:
          target['react_value'] = n_cell_period(self.id, target_id,
                                                CONF_REACT_PERIOD)
      elif target['new_sleep_end'] == 0:
        target['new_sleep_end'] = n_cell_period(self.id, target_id,
                                                period)
  [...]
```

Listing 5.8: Metodo `on_reception()` del modulo `Node`: gestione del pacchetto con periodo minimo

Successivamente, dal pacchetto vengono estrapolate varie informazioni quali l'identificativo del nodo che lo ha generato, il periodo di generazione e l'identificativo del nodo `target` a cui inoltrare il pacchetto. Se il pacchetto appena ricevuto appartiene al flusso con periodo minimo di riferimento per il link, il contatore

`sleep_end` deve essere inizializzato. Anche in questo caso viene utilizzata la funzione `n_cell_period()` per calcolare il numero di volte che le celle appartenenti al link tra questo nodo e il `target` si ripeteranno nel prossimo arco temporale di durata pari al periodo di generazione minimo. Inoltre, se è necessario attivare il meccanismo di riattivazione periodica, in questa occasione bisogna anche impostare il valore di `react_value`. Quando viene ricevuto il pacchetto con periodo minimo mentre il contatore `sleep_end` è attualmente in uso, il nuovo valore da usare nel prossimo periodo viene temporaneamente mantenuto nel contatore `new_sleep_end`.

```
1  [...]
2  if child['activation_countdown']==0
3    and child['activation_period']>0:
4      child['sleep_start'] = 1
5      child['activation_countdown'] = child['activation_period']
```

Listing 5.9: Metodo `on_reception()` del modulo `Node`: riattivazione periodica

Infine, il ricevitore controlla il valore del contatore `activation_countdown`. Nel caso in cui abbia raggiunto lo zero proprio in quella cella, il valore del contatore viene ripristinato e `sleep_start` viene impostato con il valore 1 per abilitare l'ascolto nella cella successiva.

Le funzioni `can_transmit()` e `can_receive()` sono utilizzate per verificare che il link in una data cella sia attivo rispettivamente sui nodi trasmettitore e ricevitore.

```
1 def can_transmit(self, cell: Cell) -> bool:
2   target = self.targets_status[cell.receiver_id]
3   return target['state'] != State.closed
```

Listing 5.10: Metodo `can_transmit()` del modulo `Node`

In particolare, usando le informazioni contenute in `targets_status`, il metodo `can_transmit()` controlla che il link con il nodo `target` non sia nello stato *closed*.

```
1 def can_receive(self, cell: Cell) -> bool:
2   child = self.children_status[cell.transmitter_id]
3   return child['sleep_end'] == 0 or child['sleep_start'] > 0
```

Listing 5.11: Metodo `can_receive()` del modulo `Node`

Invece, usando le informazioni contenute nella variabile `children_status`, il metodo `can_receive()` verifica che il link con il nodo figlio che intende effettuare una trasmissione in quella cella sia ancora attivo.

Capitolo 6

Risultati

Al fine di valutarne le prestazioni, gli algoritmi PRIL-MHB, PRIL-MHI e PRIL-MHI2 sono stati comparati con il TSCH Standard e con la tecnica PRIL-F.

Gli esperimenti sono stati condotti usando il simulatore descritto nel Capitolo 5, che permette di configurare facilmente diversi parametri tra cui la durata del singolo timeslot (T_{slot}), il numero di timeslot che compongono lo slotframe (N_{slot}) e il massimo numero di ritrasmissioni (N_{tries}). Inoltre, per ogni link è possibile specificare la qualità del collegamento indicando la probabilità che si verifichi un errore durante la trasmissione di un frame dati (ϵ_{data}) e durante la trasmissione di un frame di ACK (ϵ_{ack}). Per semplicità, è stata impostata la stessa probabilità di errore per tutti i link. Tutte le simulazioni sono state eseguite su un periodo temporale di un anno. I valori usati per i principali parametri di configurazione sono riportati in Tabella 6.1.

Il modello energetico usato per il simulatore è lo stesso proposto in [23] ed è stato ricavato da misurazioni eseguite su un dispositivo Open-MoteSTM con sistema operativo OpenWSN ed equipaggiato con il microcontrollore a 32 bit STM32F103RB e con il chip radio Atmel AT86RF231. Il consumo energetico in un dato timeslot è stato calcolato usando frame dati di dimensione massima (127 byte). In particolare sono stati misurati: l'energia necessaria per trasmettere un frame dati (E_{tx}), l'energia necessaria per ricevere un frame dati (E_{rx}) e l'energia consumata in idle listening (E_{idle}). Per poter essere usate nel simulatore, le misurazioni riportate in [23], che sono espresse in coulomb, sono state convertite in joule. Il valore di tensione usato per la conversione è 3 V, che corrisponde alla tensione di alimentazione tipica di un dispositivo Open-MoteSTM. La Tabella 6.2 contiene i valori dei consumi energetici usati per configurare il simulatore.

Per fornire statistiche sul consumo energetico, il simulatore calcola separatamente per ogni nodo sia la potenza totale assorbita (P_{totale}) sia il consumo energetico dovuto all'idle listening (P_{idle}). Le quantità calcolate rappresentano una media del valore della potenza valutata sull'intera durata dell'esperimento. Anche la latenza di comunicazione, che è un altro importante indice prestazionale, è stata presa in

considerazione al fine di comparare le diverse strategie proposte. Il simulatore tiene traccia del tempo che ogni pacchetto impiega per arrivare a destinazione a partire dal tempo di accodamento misurato a livello applicativo e, in questo modo, è in grado di calcolare la latenza minima (d_{min}), la latenza massima (d_{max}) e la latenza media ($\overline{d}$).

Parametro	Descrizione	Valore
T_{slot}	Durata di un timeslot	20 ms
N_{slot}	Numero di timeslot in uno slotframe	101
N_{tries}	Numero massimo di ritrasmissioni	16
ϵ_{data}	Probabilità di errore per i frame dati	0.2
ϵ_{ack}	Probabilità di errore per i frame di ACK	0.08
Q_{virt}	Parametro formula 4.1 per PRIL-MHI/PRIL-MHI2	0
Q_{real}	Parametro formula 4.1 per PRIL-MHI/PRIL-MHI2	1
$T_{riattivazione}$	Periodo di riattivazione per PRIL-MHI2	60 s

Tabella 6.1: Principali parametri di configurazione del simulatore

Quantità	Descrizione	Valore [µJ]
E_{tx}	Energia per trasmettere un frame dati e ricevere l'ACK	485.7
E_{rx}	Energia per ricevere un frame dati e trasmettere l'ACK	651.0
E_{idle}	Energia consumata in idle listening	303.3

Tabella 6.2: Modello energetico usato per le simulazioni

L'efficacia degli algoritmi è stata valutata su tre diverse configurazioni di rete: *caso a*, *caso b* e *caso c*. Il *caso a* rappresenta una rete molto piccola, utile per illustrare in modo semplice alcune peculiarità delle tecniche PRIL. Nel *caso b* vengono valutati gli effetti delle tecniche PRIL in una rete caratterizzata da periodi di generazione misti. Infine, per il *caso c* è stata usata una rete rappresentativa di casi reali. In particolare, sono stati scelti dei periodi di generazione che non sono esattamente multipli, in modo da tenere conto del fatto che la generazione dei pacchetti da parte delle applicazioni si basa su un timer locale che non è sincronizzato con quello degli altri nodi.

Per tutte le configurazioni è stata usata la stessa strategia per definire le matrici TSCH: a partire dallo *slotOffset* 1, sono state allocate prima le celle dedicate alle comunicazioni che includono i nodi foglia e successivamente sono state posizionate in maniera ordinata le celle per i nodi dei livelli superiori, partendo dal livello più lontano dalla radice fino ad arrivare a quello più vicino.

6.1 Caso a

La rete usata in questa configurazione (Figura 6.1) è molto semplice e non rispecchia un caso reale, però è utile per comprendere gli effetti che le tecniche PRIL hanno sul consumo energetico e sulle latenze. La rete è composta da cinque nodi disposti secondo una topologia ad albero a tre livelli. Nell'ultimo livello sono presenti due nodi foglia che generano periodicamente pacchetti destinati alla radice: il nodo N_1 con periodo 60 s e il nodo N_2 con periodo 120 s.

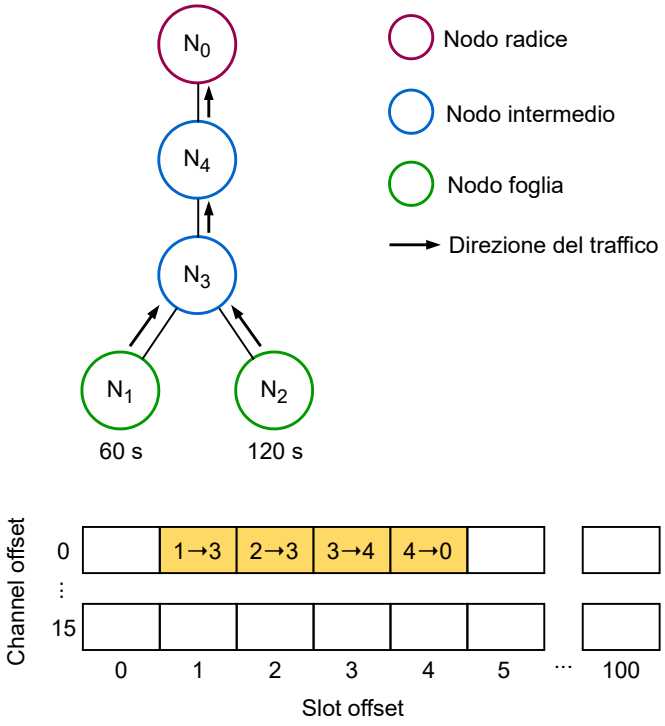


Figura 6.1: Configurazione della rete e dello schedule nel *caso a*

Come evidenziato dai risultati riportati in Tabella 6.3, la tecnica PRIL-F permette di ridurre praticamente a zero il consumo dovuto all'idle listening sul nodo N_3, ma al contempo non ha effetto sui nodi N_4 e N_0 in quanto si trovano a più di un hop di distanza dalle sorgenti dei pacchetti (N_1, N_2). Di conseguenza, il

risparmio che si riesce ad ottenere è limitato; infatti, il consumo totale della rete scende da 685.52 µW (TSCH standard) a 406.73 µW. A differenza di PRIL-F, PRIL-MH riesce a ridurre i consumi anche sui nodi oltre il primo hop, consentendo di incrementare l'efficienza energetica della rete. In questo caso, il consumo totale si riduce ulteriormente, arrivando a 143.33 µW usando PRIL-MHB e a 141.07 µW usando PRIL-MHI.

Consumo energetico [µW]

Nodo	TSCH Standard		PRIL-F		PRIL-MHB		PRIL-MHI	
	P_{idle}	P_{totale}	P_{idle}	P_{totale}	P_{idle}	P_{totale}	P_{idle}	P_{totale}
N_0	139.84	161.97	139.84	161.97	0.89	21.82	5.79	27.68
N_1	-	10.99	-	19.67	-	19.67	-	19.67
N_2	-	5.49	-	9.79	-	9.86	-	9.80
N_3	290.00	328.60	0.00088	36.84	0.00088	45.49	0.00088	38.46
N_4	139.85	178.47	139.85	178.46	0.48	46.49	5.43	45.46
Intera rete	569.69	685.52	279.69	406.73	1.37	**143.33**	11.22	**141.07**

Tabella 6.3: Comparazione dei consumi energetici tra le tecniche PRIL e TSCH Standard nel *caso a*

Latenza [s]

	TSCH Standard	PRIL-F	PRIL-MHB	PRIL-MHI
d_{min}	0.06	0.06	1.28	1.28
d_{max}	24.58	31.12	80.84	79.00
$\overline{d}$	3.45	3.45	**16.93**	**9.73**

Tabella 6.4: Comparazione dei valori di latenza tra le tecniche PRIL e TSCH Standard nel *caso a*

Confrontando i dati ottenuti per i nodi foglia (N_1, N_2), si può osservare come i loro consumi siano maggiori nelle simulazioni in cui è stata usata una delle tecniche PRIL rispetto al caso TSCH standard. Infatti, usando PRIL, può succedere che un nodo continui a trasmettere un frame dati nonostante la cella dal lato del ricevitore sia spenta. Situazioni di questo tipo si verificano quando il frame di ACK, relativo a un frame contente il comando di sleep, viene perso. Quello che accade è che il ricevitore disattiva la cella per il numero di slot indicati nel comando di sleep, mentre il trasmettitore, non avendo ricevuto l'ACK, ritenta la trasmissione. A questo punto, dato che la cella è disabilitata, il ricevitore non è in grado di ricevere il frame dati e, quindi, non può generare l'ACK, il che porta il trasmettitore a ritrasmettere lo

stesso frame fino al raggiungimento del massimo numero di trasmissioni consentite ($N_{tries} = 16$), sprecando energia. Dal momento che questo fenomeno è legato alla trasmissione del comando di sleep, esso può verificarsi solo nei nodi foglia quando viene usato PRIL-F, mentre coinvolge anche i nodi dei livelli intermedi nel caso di PRIL-MH. Infatti, osservando i risultati ottenuti per il nodo N_3, si ha che il consumo totale è maggiore per i casi PRIL-MHB e PRIL-MHI rispetto a PRIL-F. Ciononostante, l'aumento del consumo energetico dovuto alle ritrasmissioni è trascurabile se comparato al risparmio che si riesce ad ottenere grazie alle tecniche PRIL.

Lo schema in Figura 6.2 illustra la situazione appena descritta per il caso PRIL-F, riportando un esempio di comunicazione tra il nodo N_1 e il nodo N_3. L'ACK viene perso dal nodo N_1 nello slotframe numero 300, mentre il nodo N_3 disattiva la cella fino allo slotframe 329. Di conseguenza, dallo slotframe 301 al 315, il nodo N_1 effettua 15 ritrasmissioni prima che il pacchetto venga scartato.

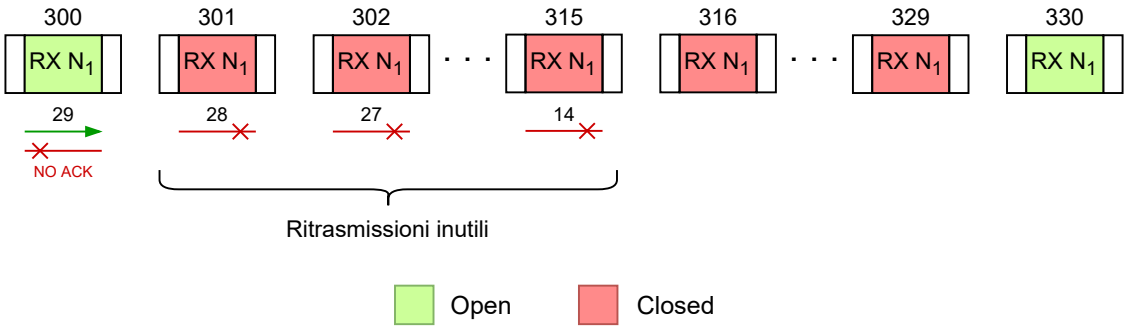


Figura 6.2: Conseguenza della perdita del frame di ACK in PRIL-F

Analizzando le latenze (Tabella 6.4) la situazione cambia. Con la tecnica PRIL-F si ottiene lo stesso valore di latenza media del TSCH standard (3.45 s in entrambi i casi), mentre si ha un peggioramento consistente usando PRIL-MH, raggiungendo il valore di 16.93 s nel caso di PRIL-MHB. Come ci si aspettava, disattivare i link sulla base dei pacchetti con periodo di generazione minimo causa rallentamenti nella consegna di quelli che vengono accodati quando la cella è già spenta. Infatti, tali pacchetti rimangono bloccati nella coda del nodo in attesa che la cella sul prossimo hop ritorni attiva. In alcuni casi, PRIL-MHI, posticipando la disattivazione del link, può impedire che la situazione appena descritta si verifichi. Un esempio è riportato in Figura 6.3, in cui le tecniche PRIL-MHB e PRIL-MHI vengono messe a confronto. Lo schema in figura illustra l'avanzamento di due pacchetti verso la destinazione (nodo N_0) a partire dal loro accodamento nel nodo N_3: il pacchetto P_1 generato dal nodo N_1, che è quello con il periodo di generazione minimo, e il pacchetto P_2 generato dal nodo N_2. Per ogni cella è riportato lo stato di attivazione del link dal lato del ricevitore.

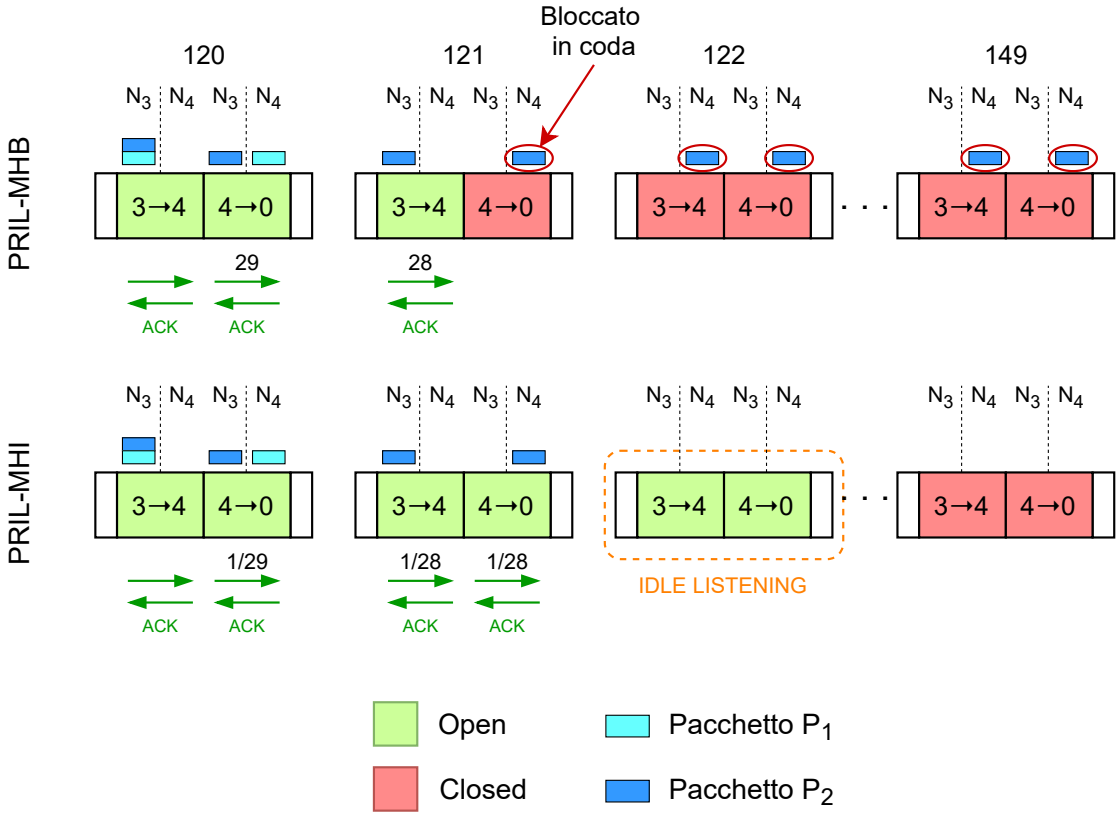


Figura 6.3: Comparazione tra PRIL-MHB e PRIL-MHI

Analizzando il caso relativo alla tecnica PRIL-MHB, si ha che durante lo slotframe 120 il nodo N_3 trasmette il pacchetto P_1 a N_4. Quest'ultimo, avendo in coda solo il pacchetto con periodo minimo, aggiunge a P_1 il comando di sleep e lo inoltra al nodo N_0. Nello slotframe successivo, N_4 riceve da N_3 anche il pacchetto P_2 che, però, rimane bloccato in coda per 29 slotframe, ossia fino alla prossima riattivazione della cella $N_4 \rightarrow N_0$.

Al contrario, usando la tecnica PRIL-MHI, il pacchetto non rimane bloccato. Infatti, durante lo slotframe 120, il nodo N_4 aggiunge al comando di sleep anche l'indicazione per il nodo N_0 di mantenere la cella attiva ancora una volta prima di disattivarla. In questo modo, quando il nodo N_4 riceve il pacchetto P_2 nello slotframe 121, la cella $N_4 \rightarrow N_0$ è ancora attiva e il pacchetto può essere trasmesso subito a N_0. Dunque, rispetto a PRIL-MHB, questa tecnica permette di ridurre la latenza media, che scende a 9.73 s, ma, come evidenziato nello schema (slotframe 122 nel caso PRIL-MHI), ha lo svantaggio di aumentare il consumo energetico causato dall'idle listening, che passa da 1.37 µW (PRIL-MHB) a 11.22 µW (PRIL-MHI).

Tuttavia, il consumo totale della rete in modalità PRIL-MHI risulta essere leggermente inferiore rispetto a PRIL-MHB. Ritardare la disattivazione di una cella

dà al trasmettitore la possibilità di effettuare eventuali ritrasmissioni, riducendo la probabilità che la situazione illustrata in Figura 6.2 si verifichi. Quindi, anche se da una parte viene incrementato il contributo in termini di energia consumata dovuto all'idle listening, dall'altra viene limitato lo spreco di energia causato dalle trasmissioni effettuate quando la cella dal lato del ricevitore è disabilitata, ottenendo una sorta di bilanciamento.

6.2 Caso b

In questa configurazione viene analizzato l'effetto delle tecniche PRIL su una rete caratterizzata da una topologia ad albero a due livelli (Figura 6.4). I nodi foglia sono configurati per generare del traffico periodico verso la radice (N_0), mentre i nodi presenti al primo livello (N_7, N_8 e N_9) hanno il compito di inoltrare i pacchetti tra le foglie e la radice. Anche in questo caso sono stati definiti dei flussi con periodi di generazione differenti: 1 min per il nodo N_1, 2 min per i nodi N_2, N_3 e N_5, 3 min per il nodo N_4 e 5 min per il nodo N_6.

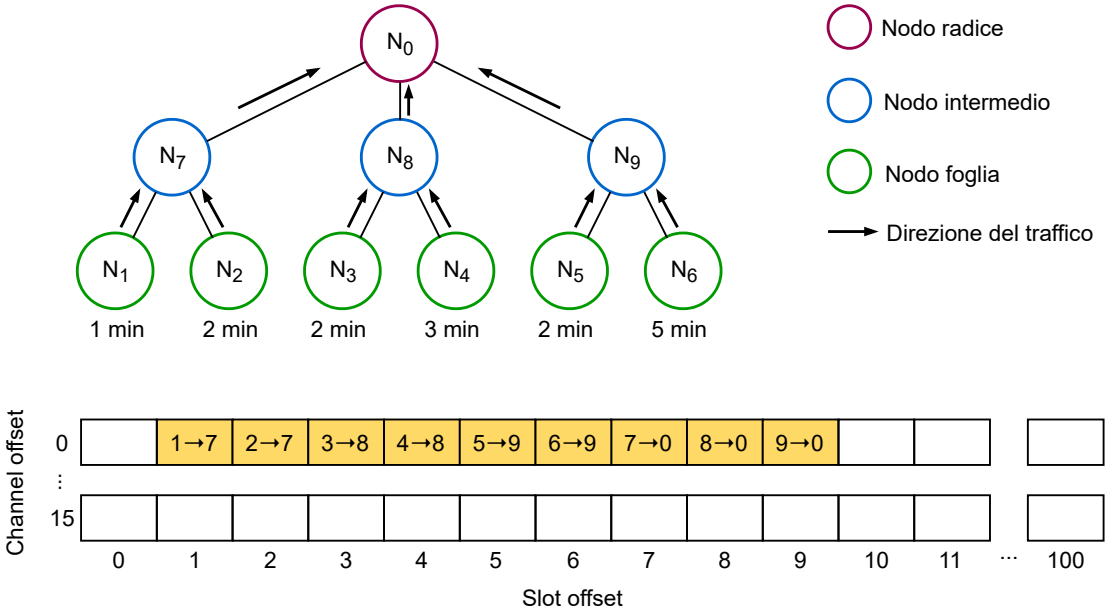


Figura 6.4: Configurazione della rete e dello schedule nel *caso b*

In Tabella 6.5 sono riportati i risultati sul consumo energetico delle quattro simulazioni, che evidenziano come il risparmio ottenuto con PRIL-F sia considerevole, poiché, agendo sul primo livello, permette di abbassare l'energia totale assorbita dai nodi N_7, N_8, N_9 di un ordine di grandezza, da 958.13 µW a 74.48 µW (riga *Livello 1* della Tabella 6.5). Purtroppo, non avendo questa tecnica alcun effetto sulla radice, il consumo della rete quando viene usato PRIL-F rimane ancora alto, passando

da 1465.82 µW a 608.34 µW. Infatti, l'energia assorbita dalla radice è significativa, dato che corrisponde a circa un terzo del consumo energetico dell'intera rete.

Consumo energetico [µW]

Nodo	TSCH Standard		PRIL-F		PRIL-MHB		PRIL-MHI		PRIL-MHI2	
	P_{idle}	P_{totale}	P_{idle}	P_{totale}	P_{idle}	P_{totale}	P_{idle}	P_{totale}	P_{idle}	P_{totale}
N_0	429.61	474.33	429.62	474.32	0.8	43.16	10.71	54.91	15.59	59.8
N_1	-	10.99	-	19.65	-	19.71	-	19.58	-	19.72
N_2	-	5.49	-	9.79	-	9.81	-	9.85	-	9.81
N_3	-	5.50	-	9.84	-	9.77	-	9.80	-	9.79
N_4	-	3.67	-	6.52	-	6.59	-	6.54	-	6.59
N_5	-	5.50	-	9.82	-	9.77	-	9.85	-	9.8
N_6	-	2.21	-	3.92	-	3.93	-	3.89	-	3.91
N_7	290.00	328.60	0.00088	36.83	0.00088	45.43	0.00088	38.52	0.00088	38.52
N_8	294.57	316.01	0.00146	20.46	0.00146	24.81	0.00146	21.33	0.00146	21.32
N_9	295.49	313.52	0.00204	17.19	0.00204	21.49	0.00204	18.02	0.00204	18.01
Livello 1	880.06	958.13	0.00438	**74.48**	0.00438	91.73	0.00438	77.87	0.00438	77.85
Intera rete	1309.67	1465.82	429.62	608.34	0.80	**194.47**	10.71	**192.29**	15.59	**197.27**

Tabella 6.5: Comparazione dei consumi energetici tra le tecniche PRIL e TSCH Standard nel *caso b*

Latenza [s]

	TSCH Standard	PRIL-F	PRIL-MHB	PRIL-MHI	PRIL-MHI2
d_{min}	0.08	0.08	0.56	0.56	0.56
d_{max}	26.18	26.56	135.40	132.92	132.62
$\overline{d}$	2.64	2.64	**11.22**	**9.93**	**5.75**

Tabella 6.6: Comparazione dei valori di latenza tra le tecniche PRIL e TSCH Standard nel *caso b*

Le tecniche PRIL-MH, riducendo il consumo dell'idle listening sul nodo N_0, permettono di ottenere degli ottimi risultati in termini di risparmio energetico. In particolare, il valore di potenza assorbita dell'intera rete scende notevolmente, arrivando a 194.47 µW nel caso di PRIL-MHB, a 192.29 µW usando PRIL-MHI e a 197.27 µW per PRIL-MHI2.

Al contrario, le latenze (Tabella 6.6) sono influenzate negativamente dagli algoritmi PRIL-MH. I valori più alti si hanno per PRIL-MHB in cui la latenza media sale da 2.64 s (TSCH standard/PRIL-F) a 11.22 s. Un leggero miglioramento può essere osservato nella simulazione di PRIL-MHI dove i pacchetti arrivano a destinazione mediamente in 9.93 s, mentre ottimi risultati si ottengono usando PRIL-MHI2 per cui il valore di latenza media è di 5.75 s.

6.3 Caso c

Nel *caso c* viene valutata l'efficacia delle tecniche PRIL sull'architettura tipica di una WSN che potrebbe essere usata per il monitoraggio di ambienti esterni. Solitamente, le reti di questo tipo sono formate da molti nodi che si occupano di trasmettere periodicamente le misurazioni di determinate condizioni ambientali a un nodo centrale (radice). Dal momento che le aree da monitorare sono spesso molto ampie, possono essere necessari più livelli di nodi intermedi per collegare i nodi lontani al nodo centrale.

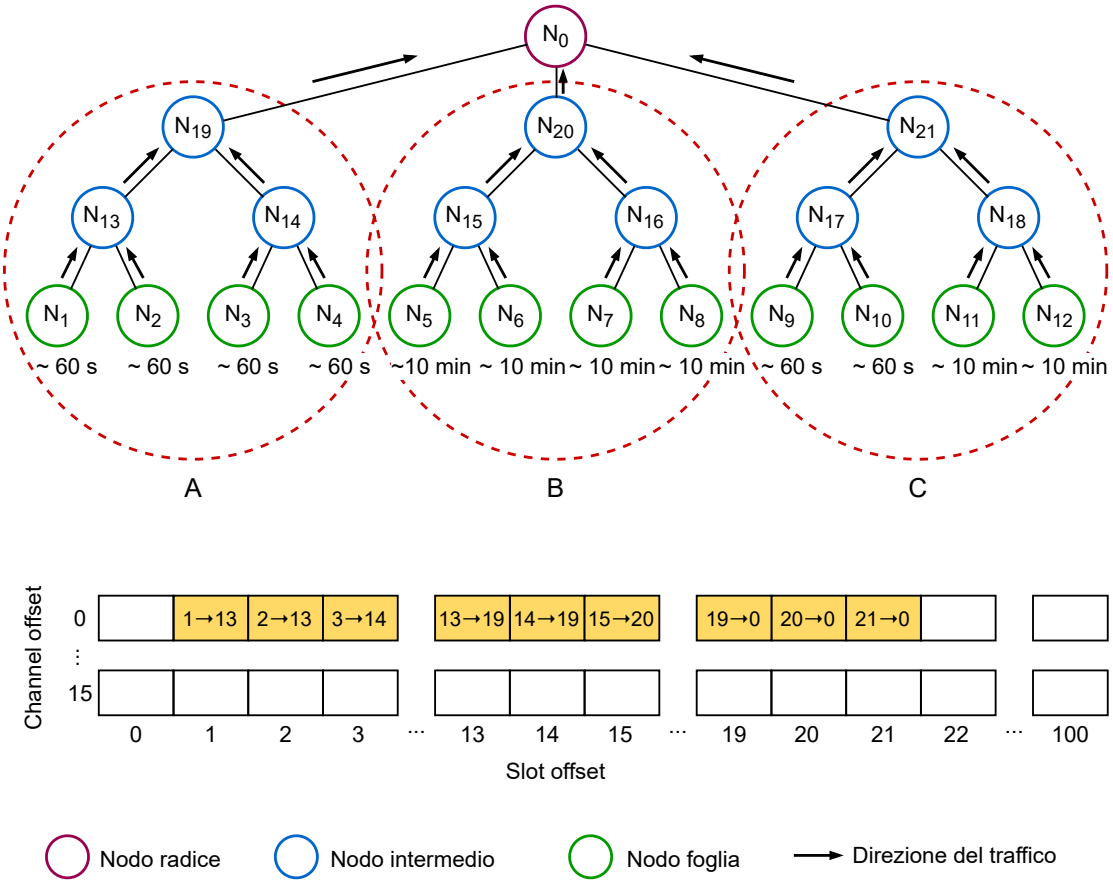


Figura 6.5: Configurazione della rete e dello schedule nel *caso c*

Escludendo la radice, la rete proposta in questa configurazione (Figura 6.5) è composta da 21 nodi, di cui 12 sono foglie e 9 sono nodi intermedi disposti su due livelli. Inoltre, facendo riferimento alla Figura 6.5, per questa rete sono stati identificati tre gruppi (A, B e C) sulla base della cadenza con cui le foglie effettuano le misurazioni e, quindi, trasmettono i pacchetti. In particolare, il gruppo A è caratterizzato da periodi di misurazione brevi (circa 60 s), i nodi del gruppo B effettuano rilevazioni con periodi più lunghi (circa 10 min), mentre nel gruppo C sono presenti

sia nodi con periodi di generazione brevi (circa 60 s) sia nodi con periodi di generazione lunghi (circa 10 min). La scelta di usare dei periodi di generazione che non sono esattamente multipli tra loro è stata fatta per tenere conto del fatto che, in contesti reali, i sensori non effettuano le misurazioni nello stesso momento a causa della non sincronizzazione a livello applicativo delle loro basi temporali. Inoltre, in questo modo si riesce ad avere maggiore randomicità sull'ordine con cui i pacchetti vengono accodati nei nodi dei livelli intermedi.

Consumo energetico [µW]

Nodo	TSCH Standard		PRIL-F		PRIL-MHB		PRIL-MHI		PRIL-MHI2	
	P_{idle}	P_{totale}	P_{idle}	P_{totale}	P_{idle}	P_{totale}	P_{idle}	P_{totale}	P_{idle}	P_{totale}
N_0	405.15	502.37	405.17	502.35	0.70	95.4	11.04	107.70	15.39	112.07
Livello 1	855.62	1025.32	855.62	1025.29	0.52	184.46	16.84	189.19	30.34	202.72
Livello 2	1756.53	1926.16	0.02	161.94	0.02	190.52	0.02	167.52	0.02	167.48
Livello 3	-	72.48	-	129.74	-	129.70	-	129.69	-	129.74
Gruppo A	846.04	1095.58	272.87	552.31	0.31	303.27	10.19	294.16	10.18	294.24
Gruppo B	895.40	920.37	297.56	325.58	0.07	30.41	1.06	29.46	10.12	38.56
Gruppo C	870.71	1008.01	285.21	439.08	0.17	171.00	5.61	162.78	10.06	167.14
Intera rete	3017.30	3526.33	1260.81	1819.32	1.24	**600.08**	27.90	**594.10**	45.75	**612.01**

Tabella 6.7: Comparazione dei consumi energetici tra le tecniche PRIL e TSCH Standard nel *caso c*

Latenza [s]

Strategia	Gruppo A			Gruppo B			Gruppo C			Intera rete		
	d_{min}	d_{max}	$\overline{d}$	d_{min}	d_{max}	$\overline{d}$	d_{min}	d_{max}	$\overline{d}$	d_{min}	d_{max}	$\overline{d}$
TSCH Standard	0.32	31.86	3.17	0.26	20.68	2.83	0.20	25.54	2.90	0.20	31.86	3.06
PRIL-F	0.32	26.88	3.17	0.26	20.14	2.84	0.20	24.96	2.90	0.20	26.88	3.06
PRIL-MHB	0.56	133.32	34.77	0.40	1206.66	**321.52**	0.30	663.52	38.30	0.30	1206.66	53.34
PRIL-MHI	0.32	127.26	32.62	0.30	1198.08	**319.09**	0.24	658.64	36.45	0.24	1198.08	51.28
PRIL-MHI2	0.32	126.84	32.63	0.26	1102.70	**57.08**	0.20	647.80	24.08	0.20	1102.70	**31.26**

Tabella 6.8: Comparazione dei valori di latenza tra le tecniche PRIL e TSCH Standard nel *caso c*

I risultati ottenuti per questa configurazione sono riassunti in Tabella 6.7. In questo caso, l'aumento di risparmio energetico che si può ottenere usando PRIL-MH al posto di PRIL-F è notevole. Infatti, nonostante quest'ultimo permetta di ridurre il consumo totale da 3526.33 µW a 1819.32 µW, usando le tecniche PRIL-MHB,

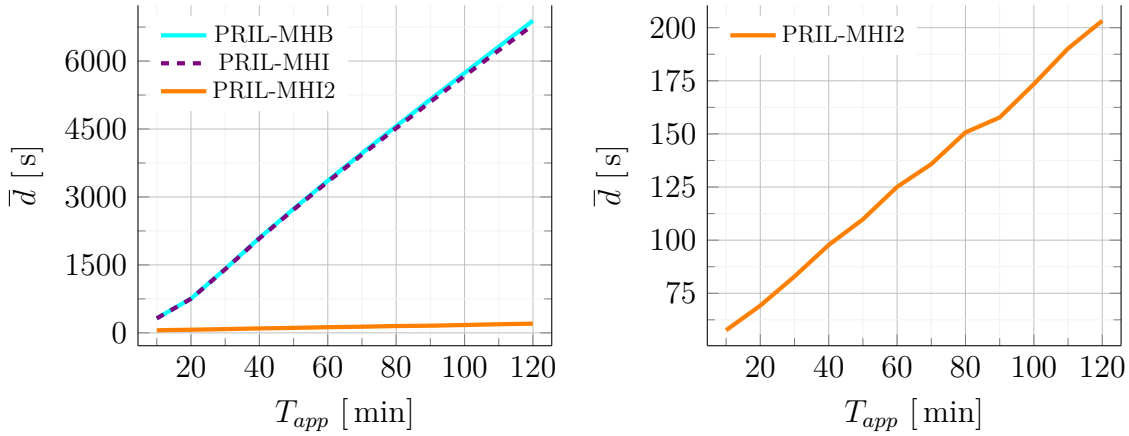
PRIL-MHI e PRIL-MHI2 si riesce a scendere rispettivamente a 600.08 µW, a 594.10 µW e a 612.01 µW.

Un'analisi più dettagliata può essere effettuata considerando singolarmente i consumi dei tre gruppi. Quello che si nota è che in PRIL-F i consumi sono bilanciati tra loro, mentre variano di molto nel caso di PRIL-MH. In particolare, facendo riferimento ai risultati di PRIL-MHB, si ha che il consumo dei nodi del gruppo A è pari a 303.27 µW, mentre per i nodi del gruppo B si riduce di un ordine di grandezza, arrivando a 30.41 µW. Ne segue che, minore è la quantità di pacchetti generati, maggiore sarà il risparmio energetico che si riesce ad ottenere.

Per quanto riguarda le latenze di trasmissione (Tabella 6.8), si ha che queste peggiorano drasticamente quando vengono usate le tecniche PRIL-MH. La latenza media passa da 3.06 s (TSCH standard/PRIL-F) a 53.34 s in PRIL-MHB e 51.28 s in PRIL-MHI. Dai risultati ottenuti per questa rete, risulta evidente che nelle strategie PRIL-MH le latenze sono strettamente legate ai periodi di generazione dei pacchetti e al numero di link che un pacchetto attraversa per arrivare a destinazione. Infatti, nel peggiore dei casi, ogni link può introdurre un ritardo al tempo di arrivo di un pacchetto di una quantità pari al minimo tra i periodi di generazione dei pacchetti che lo attraversano. Questo accade quando un nodo intermedio riceve un pacchetto subito dopo la disattivazione della cella sul prossimo hop. Per questo motivo i valori di latenza più alti si hanno per il gruppo B, in cui il periodo di generazione minimo di riferimento per tutti i link è di 10 min e, quindi, i pacchetti possono rimanere bloccati 10 min nella coda di un nodo del secondo livello (N_{15} o N_{16}) e ulteriori 10 min nella coda di N_{20}. Infatti, i valori di latenza massima per il gruppo B nei casi PRIL-MHB e PRIL-MHI sono rispettivamente di 1206.66 s e di 1198.08 s, ossia circa 20 min per entrambe le strategie. I valori di latenza più bassi si ottengono per il gruppo A dove tutti i nodi foglia generano pacchetti ogni circa 60 s e, pertanto, la latenza massima è 133.32 s per PRIL-MHB e 127.26 s per PRIL-MHI. Invece, i risultati relativi al gruppo C si posizionano a metà tra quelli del gruppo A e quelli del gruppo B: 38.30 s usando PRIL-MHB e 36.45 s usando PRIL-MHI. In questo caso, i pacchetti generati dal nodo N_{12} possono rimanere bloccati fino a 10 min nella coda del nodo N_{18} e fino a 60 s nella coda del nodo N_{21}.

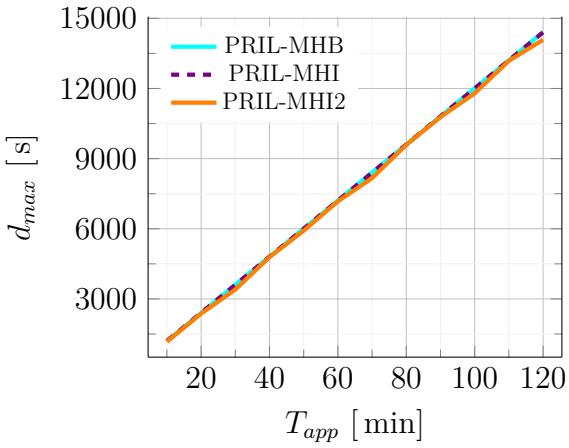
Notevoli miglioramenti sulla latenza media possono essere ottenuti usando PRIL-MHI2. Questa tecnica permette di impostare la riattivazione periodica di un link per dare la possibilità ad un nodo di trasmettere eventuali pacchetti che si sono accodati mentre la cella sul prossimo hop è spenta. Il periodo di riattivazione scelto ($T_{riattivazione} = 60$ s) consente di ridurre il valore della latenza media fino a 31.26 s. I benefici maggiori si osservano nel gruppo B in cui la latenza media cala drasticamente arrivando a 57.08 s. Piccoli miglioramenti sono apprezzabili anche nel gruppo C in cui la riattivazione periodica ha effetto solo sul link $N_{18} \rightarrow N_{21}$. Invece, nel gruppo A la riattivazione periodica non viene impostata per nessun link in quanto il periodo di generazione minimo di riferimento per tutti i link è di 60 s. Per questo motivo, non ci sono differenze tra PRIL-MHI e PRIL-MHI2.

Usando PRIL-MHI2, però, non si ha nessun miglioramento sulla latenza massima (d_{max}). Infatti, quando il comando di riattivazione periodica non va a buon fine, i pacchetti possono rimanere bloccati nella coda di un nodo intermedio, in modo analogo a quello che accade in PRIL-MHB e PRIL-MHI.



(a) Latenza media

(b) Latenza media PRIL-MHI2



(c) Latenza massima

Figura 6.6: Latenza media (in alto a sinistra) e massima (in basso) delle tecniche PRIL-MHB, PRIL-MHI e PRIL-MHI2 al variare di T_{app}. In figura è stato anche incluso il grafico ingrandito della latenza media per la tecnica PRIL-MHI2 (in alto a destra)

Ulteriori simulazioni sono state eseguite al fine di osservare le variazioni dei valori di latenza media ($\overline{d}$) e di latenza massima (d_{max}) in relazione al periodo di generazione minimo per le tecniche PRIL-MHB, PRIL-MHI e PRIL-MHI2. Per questa analisi è stata utilizzata la stessa rete del *caso c*, ma sono stati cambiati i

periodi di generazione. In particolare, a tutti i nodi foglia sono stati assegnati dei periodi di generazione circa pari a un periodo di riferimento T_{app}. La simulazione, la cui durata è stata impostata su un anno, è stata ripetuta 12 volte per ogni strategia. Il valore di T_{app} è stato fatto variare ad intervalli di 10 min, iniziando da $T_{app} = 10\,\text{min}$ e terminando a $T_{app} = 120\,\text{min}$.

I risultati ottenuti sono stati riportati nei grafici in Figura 6.6. In particolare, la Figura 6.6a mostra l'andamento dei valori della latenza media per le strategie PRIL-MHB, PRIL-MHI e PRIL-MHI2, mentre in Figura 6.6b è riportato solo quello della tecnica PRIL-MHI2 con lo scopo di renderne maggiormente visibile la variazione. Invece, i valori di latenza massima sono riportati nel grafico in Figura 6.6c.

I grafici evidenziano come, indipendentemente dalla tecnica usata, la latenza e il periodo di generazione (T_{app}) siano legati da una relazione lineare. Inoltre, si osserva che la crescita della latenza media nel caso di PRIL-MHI2 è decisamente più lenta rispetto alle tecniche PRIL-MHB e PRIL-MHI, per le quali, invece, l'andamento è pressoché identico. Tuttavia, per quanto riguarda la latenza massima, si può affermare che PRIL-MHI2 risulta essere equivalente a PRIL-MHB e a PRIL-MHI.

Capitolo 7

Conclusioni

In questa tesi sono state proposte tre versioni della tecnica PRIL-MH volte alla riduzione del fenomeno di idle listening in reti di sensori wireless con topologie ad albero multi-livello. Per poterle facilmente confrontare con PRIL-F e TSCH Standard, tali tecniche sono state implementate nel simulatore `TSCH-predictor`. Le metriche prese in considerazione per la valutazione sono la latenza minima (d_{min}), la latenza massima (d_{max}), la latenza media ($\overline{d}$), il consumo energetico totale (P_{totale}) e il contributo dovuto all'idle listening (P_{idle}).

Le simulazioni evidenziano che, usando il TSCH Standard, lo spreco di energia dovuto all'idle listening è considerevole, rappresentando circa l'80% del consumo totale della rete. Nonostante la tecnica PRIL-F permetta di ridurre consistentemente il consumo sui nodi della rete al primo hop, la quantità totale di energia sprecata in idle listening è ancora considerevole. Infatti, i risultati relativi alle tecniche PRIL-MH mostrano una drastica diminuzione del consumo energetico su tutti i nodi della rete, soprattutto nelle simulazioni effettuate su una rete a più livelli, in cui la quantità totale di energia sprecata a causa dell'idle listening si riduce fino a tre ordini di grandezza rispetto a PRIL-F.

Purtroppo, l'utilizzo delle tecniche PRIL-MH porta a un peggioramento delle latenze di comunicazione. Infatti, per ogni link che un pacchetto deve attraversare, le tecniche PRIL-MH potrebbero introdurre un ritardo proporzionale al periodo minimo di riferimento usato per la disattivazione del link. Tuttavia, le latenze rivestono solo un ruolo marginale nella maggior parte delle applicazioni in cui le reti di sensori wireless sono usate. Infatti, il requisito principale delle WSN è il risparmio energetico e, sotto questo aspetto, tutte e tre le versioni di PRIL-MH permettono di ottenere degli ottimi risultati. Ad ogni modo, la tecnica PRIL-MHI2 rappresenta una valida soluzione anche per i casi in cui è fondamentale ridurre gli sprechi energetici senza, però, peggiorare drasticamente le latenze.

Lavori futuri potrebbero concentrarsi sul perfezionamento della tecnica PRIL-MH, implementando degli algoritmi intelligenti che siano in grado di dedurre autonomamente il tipo di traffico (periodico o sporadico) e di prevedere il tempo di

accodamento dei pacchetti. In questo modo non sarebbe più necessario aggiungere l'informazione sul periodo di generazione esplicitamente nel frame dati, riducendo così la dimensione dei pacchetti trasmessi. Inoltre, i nodi potrebbero raccogliere delle statistiche sui link in modo che sia possibile calcolare i valori statisticamente migliori per la durata e l'inizio del periodo di disattivazione del link.

Bibliografia

[1] IEEE Standard for Low-Rate Wireless Networks. *IEEE Std 802.15.4-2020 (Revision of IEEE Std 802.15.4-2015)*, pages 1–800, 2020.

[2] IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer. *IEEE Std 802.15.4e-2012 (Amendment to IEEE Std 802.15.4-2011)*, pages 1–225, 2012.

[3] Stefano Scanzio, Gianluca Cena, Adriano Valenzano, and Claudio Zunino. Energy Saving in TSCH Networks by Means of Proactive Reduction of Idle Listening. In Luigi Alfredo Grieco, Gennaro Boggia, Giuseppe Piro, Yaser Jararweh, and Claudia Campolo, editors, *Ad-Hoc, Mobile, and Wireless Networks*, pages 131–144, Cham, 2020. Springer International Publishing.

[4] Mashood Anwar, Yuanqing Xia, and Yufeng Zhan. TDMA-Based IEEE 802.15.4 for Low-Latency Deterministic Control Applications. *IEEE Transactions on Industrial Informatics*, 12(1):338–347, 2016.

[5] Stefano Scanzio, Lukasz Wisniewski, and Piotr Gaj. Heterogeneous and dependable networks in industry – A survey. *Computers in Industry*, 125:103388, 2021.

[6] Diwaker Pant, Sandeep Verma, and Piyush Dhuliya. A study on disaster detection and management using WSN in Himalayan region of Uttarakhand. In *2017 3rd International Conference on Advances in Computing,Communication Automation (ICACCA) (Fall)*, pages 1–6, 2017.

[7] Lim Yong Li, Haryati Jaafar, and Nur Hidayah Ramli. Preliminary Study of Water Quality Monitoring Based on WSN Technology. In *2018 International Conference on Computational Approach in Smart Systems Design and Applications (ICASSDA)*, pages 1–7, 2018.

[8] Abbas Javed, Hadi Larijani, Ali Ahmadinia, and Des Gibson. Smart Random Neural Network Controller for HVAC Using Cloud Computing Technology. *IEEE Transactions on Industrial Informatics*, 13(1):351–360, 2017.

[9] Chaitanya Vijaykumar Mahamuni. A military surveillance system based on wireless sensor networks with extended coverage life. In *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*, pages 375–381, 2016.

[10] Bhanumathi Velusamy and Sangeetha Chitteth Pushpan. An Enhanced Channel Access Method to Mitigate the Effect of Interference Among Body Sensor Networks for Smart Healthcare. *IEEE Sensors Journal*, 19(16):7082–7088, 2019.

[11] Jiangyu Yan, Ran Qiao, Liangrui Tang, Chenxi Zheng, and Bing Fan. A fuzzy decision based WSN localization algorithm for wise healthcare. *China Communications*, 16(4):208–218, 2019.

[12] Nagwan M. Abdelsamee, Hanoof Alsulaimani, and Nouf Albatati. Experimental Setup of Smart Traffic Control using Wireless Sensor Networks. In *2018 21st Saudi Computer Society National Computer Conference (NCC)*, pages 1–7, 2018.

[13] Stefano Scanzio, Mohammad Ghazi Vakili, Gianluca Cena, Claudio Giovanni Demartini, Bartolomeo Montrucchio, Adriano Valenzano, and Claudio Zunino. Wireless Sensor Networks and TSCH: A Compromise Between Reliability, Power Consumption, and Latency. *IEEE Access*, 8:167042–167058, 2020.

[14] Atis Elsts, Seohyang Kim, Hyung-Sin Kim, and Chongkwon Kim. An Empirical Survey of Autonomous Scheduling Methods for TSCH. *IEEE Access*, 8:67147–67165, 2020.

[15] Lucia Lo Bello, Alfio Lombardo, Sebastiano Milardo, Gaetano Patti, and Marco Reno. Software- Defined Networking for Dynamic Control of Mobile Industrial Wireless Sensor Networks. In *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 290–296, 2018.

[16] Maria Rita Palattella, Thomas Watteyne, Qin Wang, Kazushi Muraoka, Nicola Accettura, Diego Dujovne, Luigi Alfredo Grieco, and Thomas Engel. On-the-Fly Bandwidth Reservation for 6TiSCH Wireless Industrial Networks. *IEEE Sensors Journal*, 16(2):550–560, 2016.

[17] Gianluca Cena, Stefano Scanzio, Lucia Seno, Adriano Valenzano, and Claudio Zunino. Energy-Efficient Link Capacity Overprovisioning In Time Slotted Channel Hopping Networks. In *2020 16th IEEE International Conference on Factory Communication Systems (WFCS)*, pages 1–8, 2020.

[18] Maurizio Mongelli and Stefano Scanzio. A neural approach to synchronization in wireless networks with heterogeneous sources of noise. *Ad Hoc Networks*, 49:1–16, 2016.

[19] J.R. Vig. Quartz Crystal Resonators and Oscillators - For Frequency Control and Timing Applications - A Tutorial. US Army Communications-Electronics Research, Development and Engineering Center Fort Monmouth, NJ, USA, Rev. 8.5.2.2, 2004.

[20] Stefano Scanzio, Gianluca Cena, Lucia Seno, and Adriano Valenzano. Robustness and Optimization of PRIL Techniques for Energy Saving in TSCH Networks. In *2021 IEEE 26th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8, Sweden, 2021.

[21] Esteban Municio, Glenn Daneels, Mališa Vučinić, Steven Latré, Jeroen Famaey, Yasuyuki Tanaka, Keoma Brun, Kazushi Muraoka, Xavier Vilajosana, and Thomas Watteyne. Simulating 6TiSCH networks. *Transactions on Emerging Telecommunications Technologies*, 30(3):e3494, 2019.

[22] Atis Elsts. TSCH-Sim: Scaling Up Simulations of TSCH and 6TiSCH Networks. *Sensors*, 20(19):5663, 2020.

[23] Xavier Vilajosana, Qin Wang, Fabien Chraim, Thomas Watteyne, Tengfei Chang, and Kristofer S. J. Pister. A Realistic Energy Consumption Model for TSCH Networks. *IEEE Sensors Journal*, 14(2):482–489, 2014.