Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea in Ingegneria dell'Informazione

Tesi di Laurea Triennale

Algoritmi di rate adaptation per dispositivi LoRa privi di batteria

Candidato
Alessia Ortile
1229242

Relatore
Prof. Andrea Zanella
Correlatore
Martina Capuzzo

Anno Accademico 2021/2022

Sommario

Nei sistemi Internet of Things (IoT) sono presenti migliaia di dispositivi usati per l'acquisizione di dati dall'ambiente che permettono di implementare diversi tipi di applicazioni in scenari sia urbani che non. La sostituzione delle batterie di questi dispositivi comporta un elevato costo economico e ambientale: pertanto, si stanno studiando soluzioni caratterizzate dall'assenza di batteria e alimentate da fonti energetiche rinnovabili.
Una delle tecnologie di comunicazioni per reti IoT più promettenti e diffuse è LoRaWAN. Questa tecnologia permette di implementare comunicazioni a lungo raggio con un basso consumo energetico sfruttando frequenze pubbliche. Questi fattori la rendono la più indicata per applicazioni IoT dove il risparmio energetico è un elemento cruciale. La tesi presenta l'implementazione e l'analisi, attraverso il simulatore ns-3, di un algoritmo di Adaptive Data Rate per reti LoRaWAN i cui nodi, privi di batteria, sono alimentati solamente da un pannello solare.

Indice

Capitolo 1

Introduzione

Internet of Things (IoT) nel mondo delle telecomunicazioni è un nuovo sistema di comunicazione, l'evoluzione del concetto di Internet, pensato per adattarsi ad una serie vasta ed eterogenea di elementi che prendono parte alla comunicazione. Le applicazioni di questo paradigma sono svariate: nel campo sanitario permettono di monitorare da remoto pazienti costretti a casa a causa di una condizione di scarsa mobilità o dell'età avanzata; l'emergenza sanitaria causata dal COVID-19 ha ulteriormente diffuso queste pratiche. Nel campo dell'agricoltura, reti di sensori in comunicazione tra loro permettono di aggiornare in tempo reale l'agricoltore sullo stato dei raccolti e del terreno. Questo permette l'utilizzo di sistemi di irrigazione e fertilizzazione automatizzati e solo al bisogno, riducendo l'utilizzo di queste risorse. In ambienti urbani sono utilizzate dalle amministrazioni comunali per affrontare sfide come la riduzione dei consumi, del traffico e delle emissioni. Alcuni esempi sono i parcheggi che notificano attraverso un'applicazione se sono occupati, i cestini che notificano alla società di raccolta il livello di riempimento o i semafori che in base alle condizioni del traffico regolano la loro accensione.
Questo mercato è cresciuto considerevolmente negli ultimi anni, è stato stimato che nel 2030 potrebbe valere dai 5.5 ai 12.6 trilioni di dollari, di cui 970-1700 bilioni dipendono solamente dalle *Smart Cities* [1], cioè dalle città che sfruttano sistemi digitali come quelli citati sopra per migliorare l'efficienza delle loro infrastrutture.

Le applicazioni della tecnologia IoT nel settore urbano e agricolo necessitano di molti sensori, distribuiti in una vasta area, che comunicano tra loro. Per questo le tecnologie che le implementano devono essere scalabili, economiche e devono permettere comunicazioni a lungo raggio. Inoltre, gioca un ruolo importante l'efficienza energetica di queste tecnologie:

la maggior parte dei sensori infatti viene alimentato attraverso una batteria, ma questo comporta una serie di problemi:

a) Spesso la sostituzione delle batterie comporta un costo economico molto elevato a causa della forza lavoro richiesta, delle posizioni non sempre accessibili dei sensori e del costo delle batterie.

b) Il malfunzionamento di un sensore a causa dello scaricamento della batteria può portare ad un danno economico in alcune applicazioni.

c) Le batterie non sono riciclabili: data la diffusione di queste tecnologie la loro sostenibilità non è più un fattore marginale per l'ambiente.

Le soluzioni utilizzate attualmente cercano di allungare la vita delle batterie in due modi: a livello hardware migliorando l'efficienza, e a livello software riducendo il numero di pacchetti inviati e pianificando periodi di accensione e spegnimento. In minor numero sono le soluzioni che eliminano completamente la batteria e si affidano esclusivamente all'energia raccolta dall'ambiente. Queste soluzioni sono quelle su cui ci concentreremo in questa tesi.

Tra le tecnologie di comunicazione tuttora esistenti, la classe Low Power Wide Area Network (LPWAN) rispecchia le specifiche richieste dalle comunicazioni IoT. In questa categoria, LoRaWAN (Long Range Wide Area Network) è una tra le più promettenti, anche se molte tecniche per migliorare la sua scalabilità sono ancora in via di sviluppo.
La soluzione LoRaWAN si può dividere in due componenti principali: la modulazione LoRa e il protocollo LoRaWAN.
LoRa descrive la modulazione del segnale fisico, la tecnologia che usa è proprietaria di Semtech. Le specifiche a livello MAC dipendono dal protocollo LoRaWAN, che è aperto e permette di sfruttare le proprietà di LoRa nel migliore modo possibile.
Inoltre, LoRaWAN è un ottimo candidato per l'implementazione di sistemi IoT che mirano a ridurre il loro impatto ambientale. Infatti, le comunicazioni a lungo raggio ma a bassa potenza permettono di limitare l'uso di batterie a favore di alternative meno inquinanti che sfruttano l'energia prodotta da fonti rinnovabili.
Per questa ragione questa tesi si focalizza sullo studio dello standard LoRaWAN e del suo utilizzo in reti formate da dispositivi privi di batteria. Il resto della tesi è sviluppato come segue.

Nel capitolo 1 si presenta una panoramica delle applicazioni dei sistemi IoT. Il capitolo 2 descrive la modulazione LoRa e lo standard LoRaWAN, nel capitolo 3 si presenta lo stato dell'arte degli algoritmi di adapitve data rate analizzandone difetti e pregi. Inoltre, presenta una panoramica della struttura e del funzionamento dei dispositivi privi di batteria. Il simulatore di rete usato viene introdotto nel capitolo 4, seguito dalla spiegazione della struttura del modulo LoRaWAN e del contributo apportato ad esso. Il capitolo 5 discute ed analizza i risultati ottenuti, il capitolo 6 contiene le conclusioni tratte dal lavoro fatto.

Capitolo 2

LoRaWAN

2.1 LoRa

LoRa (Long Range) è una tecnica di modulazione di livello fisico basata su una tecnologia *chirp spread spectrum* ideata da Semtech Corporation [2]. I segnali usati variano la loro frequenza linearmente nel tempo, la durata di ciascun simbolo modulato dipende da un parametro chiamato Spreading Factor (SF) che assume valori nell'intervallo [7,12]. Spreading Factor maggiori impiegano un tempo maggiore per inviare un simbolo. In particolare, aumentare lo SF di 1 comporta raddoppiare il tempo di simbolo. Di contro, SF maggiori sono caratterizzati da una sensitività al ricevitore minore rispetto a SF più elevati, quindi il segnale trasmesso può essere ricevuto correttamente a distanze più ampie.

Il resto del capitolo descrive in maggior dettaglio le caratteristiche dello SF, introduce lo standard LoRaWAN e illustra l'utilità e il funzionamento degli algoritmi di ADR.

2.1.1 Spreading Factor

Uno dei parametri che la modulazione LoRa permette di configurare è lo SF. Esso determina, insieme alla banda usata (B), il tempo necessario per l'invio di un simbolo (T_s).

$$T_s = \frac{2^{SF}}{B} \tag{2.1}$$

Si osserva che incrementando lo SF di uno, T_s raddoppia; l'aumento di T_s comporta un uso maggiore di energia per simbolo, quindi un segnale più robusto. Per queste motivazioni

SF minori non sono adatti a trasmissioni a lungo raggio. La scelta dello SF da utilizzare dipende dal Received Signal Strength Indicator (RSSI) medio con cui i segnali raggiungono i Gateways (GW) (vedi tabella 2.1).

SF	RSSI [dBm]	Distance [m]	DR [bit/s]
7	-126.50	74	5470
8	-127.25	78	3125
9	-131.25	94	1760
10	-132.75	101	980
11	-134.50	110	440
12	-133.25	104	250

Tabella 2.1: Soglie di sensibilità e distanza raggiunta calcolata con il modello path loss specificato in [3] con larghezza di banda pari a 125kHz

D'altro canto, l'incremento del valore dello SF porta a un *Time-on-Air* (ToA) maggiore. ToA è definito uguale a $T_s \cdot PL_{sym}$ dove PL_{sym} è la dimensione del payload espressa in simboli, e corrisponde al tempo di trasmissione del pacchetto.
Occupare il canale a lungo aumenta il rischio di collisioni tra i pacchetti trasmessi, ma se lo stesso canale viene occupato da segnali con SF differenti le collisioni non si verificheranno. Questo dipende dal fatto che in LoRa segnali modulati con SF diversi sono ortogonali tra loro [2].
Infine, lo SF influenza il data rate (DR): esso diminuisce all'aumentare del valore dello SF. Nella tabella 2.1 sono presenti i valori di DR associati ad ogni SF.

$$DR = SF \cdot \frac{B}{2^{SF}} \tag{2.2}$$

Da queste proprietà si può evincere che la scelta di come assegnare gli SF gioca un ruolo fondamentale per reti LoRaWAN su larga scala.

2.2 Standard LoRaWAN

Il protocollo LoRaWAN caratterizza la comunicazione tra dispositivi LoRa a livello MAC; è stato sviluppato da LoRa Alliance, la documentazione che lo descrive è pubblica [4].

2.2.1 Architettura e dispositivi

Le specifiche LoRaWAN definiscono tre componenti principali che formano l'architettura di rete:

End device (ED): dispositivi a bassa potenza che comunicano con uno o più gateways usando la modulazione LoRa e il protocollo LoRaWAN. Di solito sono dotati di sensori per raccogliere e monitorare i dati periodicamente;

Gateways (GW): dispositivi intermedi che fanno da ponte tra i nodi periferici e la rete. Inoltrano i pacchetti ricevuti tramite LoRa al Network Server attraverso Internet o una rete privata;

Network Server (NS): è il controllore centrale della rete, è responsabile di scartare i pacchetti duplicati, decodificare i pacchetti ricevuti dai nodi e generare i pacchetti di risposta.

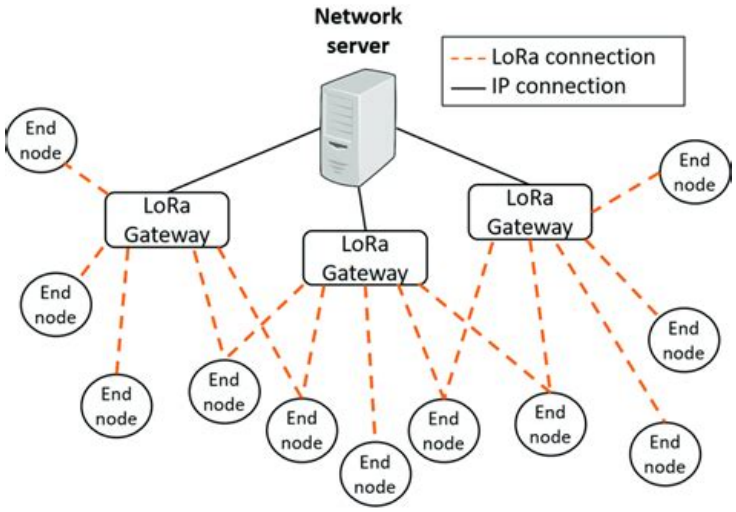

Figura 2.1: Topologia di una rete LoRaWAN

Come mostrato in figura 2.1, la rete ha una topologia a stella e gli EDs non sono associati ad un unico GW ma a tutti quelli raggiungibili. I pacchetti ricevuti dai GWs sono inoltrati al NS con l'aggiunta di informazioni legate all'affidabilità del collegamento misurato tramite il valore di Signal-To-Noise Ratio (SNR) o RSSI. Sarà compito del NS verificare l'autenticità e integrità dei pacchetti ricevuti, scartare i duplicati, selezionare il GW a cui mandare la risposta ed eventualmente inviare i dati ricevuti agli Application Server

opportuni che si occuperanno della loro analisi.

LoRaWAN definisce tre tipi di classi di End Device supportate:

- **Classe A (All):** Gli ED possono decidere autonomamente quando trasmettere un pacchetto, ogni trasmissione è seguita dall'apertura di due brevi finestre di ricezione downlink. Qualsiasi comunicazione da NS a ED deve aspettare l'apertura di queste finestre: questo li rende i dispositivi con il minore consumo energetico, ma ha lo svantaggio di introdurre latenza per i messaggi dal NS verso l'ED.
- **Classe B (Beacon):** Oltre alle funzionalità dei dispositivi di classe A, i dispositivi di classe B aprono una finestra di ricezione aggiuntiva a un tempo predefinito. La sincronizzazione tra NS e ED avviene grazie all'uso di un messaggio di beacon.
- **Classe C (Continuosly listening):** Questi dispositivi sono quelli con il maggiore consumo energetico perché sono sempre in ricezione tranne quando stanno trasmettendo.

In questa tesi è stato scelto di utilizzare solo ED di classe A dato che il loro basso consumo energetico è funzionale al caso di studio.

2.2.2 Finestra di ricezione

Ogni classe di dispositivi citati è caratterizzata da differenti modalità di ricezione e invio dei dati. Nel seguito si descrivono in dettaglio quelle di classe A.
Per la maggior parte del tempo questi dispositivi sono in stato di sleep per minimizzare il consumo energetico. L'invio di un pacchetto avviene ogni qual volta il livello applicazione lo richiede e porta il dispositivo in fase di trasmissione (TX). Un secondo dopo la fine della trasmissione si apre la prima finestra di ricezione (RX1) con frequenza e data rate pari a quelli di invio. Se nessuna risposta viene ricevuta nella prima finestra, si apre la seconda (RX2).
RX2 ha frequenza e data rate fissati, i cui valori possono essere configurati a livello MAC. Di default vengono assegnati in modo tale da massimizzare le probabilità di ricezione del messaggio, quindi viene scelto SF pari a 12. Di contro questa scelta comporta una maggiore durata di RX2 e quindi un maggiore consumo energetico.
Scegliere di usare gli stessi valori usati in RX1 può ridurre sensibilmente questo consumo senza degradare il tasso di ricezione [5].

La durata delle finestre di ricezione può essere variabile, ma deve essere sempre di una durata sufficiente per ricevere il preambolo del pacchetto; se questa rilevazione avviene, il dispositivo rimarrà in stato di ricezione fino alla fine del messaggio.

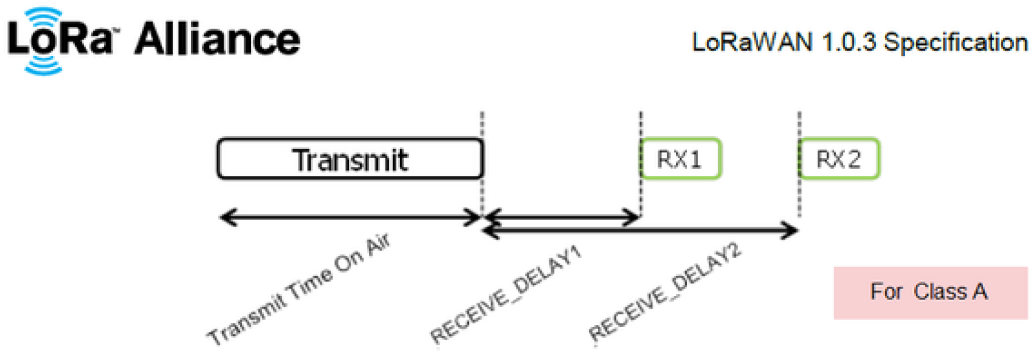

Figura 2.2: Finestra di ricezione

2.2.3 Algoritmi di Adaptive Data Rate

Gli algoritmi di Adaptive Data Rate (ADR) si occupano di definire per ogni ED un opportuno SF da usare per trasmettere i pacchetti.
Si dividono in *Network-Managed ADR* e *Blind ADR*. I primi sono eseguiti dal NS in funzione dei dati ricavati dai pacchetti ricevuti, come SNR e RSSI.
Nel caso di nodi mobili, invece, un algoritmo centralizzato non è consigliabile a causa delle continue variazioni dell'attenuazione del canale, quindi vengono usati i *Blind ADR* eseguiti direttamente dall'ED.
L'algoritmo Network-Managed di base assegna SF in funzione del parametro RSSI, quindi sceglie il valore minimo che assicuri la ricezione del messaggio.
Date le proprietà degli SF viste nella sezione 2.1.1, è facile osservare che questo algoritmo non tiene in considerazione molte di esse, per esempio l'ortogonalità delle diverse modulazioni. Per questo sono state sviluppate strategie più articolate per cercare di ottenere l'assegnazione che massimizzi le prestazioni della rete. Nel prossimo capitolo ne sono descritte alcune che attraverso metodi diversi migliorano fattori come il throughput o la durata della batteria.

Capitolo 3

Stato dell'arte

3.1 Algoritmi di ADR

Il protocollo LoRaWAN definisce l'accesso al canale in modalità ALOHA, questo rende le trasmissioni concorrenti che interferiscono tra loro uno dei principali problemi legato a questo tipo di reti.
Questo problema può essere affrontato in vari modi, ossia attraverso metodi di cancellazione delle interferenze del segnale dopo una collisione [6], intensificando in maniera opportuna la presenza di GWs nella rete, oppure attraverso una diversa assegnazione degli SF. Le politiche di assegnazione non si limitano solamente a ridurre le collisioni: alcune mirano a minimizzare il consumo energetico, o a massimizzare il data rate, oppure a minimizzare le interferenze.
Per esempio, *EXPLoRa-SF* associa ad ogni SF lo stesso numero di ED, agli ED caratterizzati da valori di RSSI migliori vengono assegnati SF minori [7]. Un altro algoritmo, invece, decide come suddividere gli ED in modo da equalizzare la probabilità di successo per ogni SF invece che assegnare sempre lo stesso numero di dispositivi [8].
EXPLoRa-K cambia completamente approccio: prima identifica le aree particolarmente critiche, quelle dove la concentrazione di ED è alta, poi monitora le collisioni in questa area. Se le collisioni associate ad un nodo sono troppo frequenti, il suo SF viene incrementato [3]. Un'altra tecnica assegna a differenti EDs lo stesso SF per uno slot di tempo limitato. Ogni ED comunica solo durante l'intervallo di tempo assegnato. Un approccio basato sulla teoria dei giochi permette di trovare la configurazione che massimizza il numero di pacchetti correttamente ricevuti e minimizza il tempo di attesa medio della rete [9].

Esiste una versione più sofisticata di *EXPLoRa-SF*, chiamata *EXPLoRa-AT*, che permette di migliorare il Data Extraction Rate (DER), soprattutto in situazioni di carico di rete elevato. Il suo funzionamento si basa sull'equalizzare il tempo per cui ogni gruppo di ED con lo stesso SF usa il canale (Time-on-Air). Per farlo assegna ad ogni canale un numero massimo di nodi che possono usarlo; i nodi saranno distribuiti attraverso un approccio "*ordered water-filling*" tenendo conto delle variazioni del Time-on-Air (ToA) in base allo SF usato [7]. La sua applicazione è limitata da due ipotesi:

- La dimensione dei pacchetti è costante per tutti gli ED e per tutte le trasmissioni;
- La frequenza di invio dei pacchetti è costante nel tempo e costante tra tutti gli ED;

Per cercare di sopperire a queste limitazioni è stato pensato EXPLoRa-ST.

3.1.1 Explora-ST

EXPLoRa-ST si pone l'obiettivo di migliorare EXPLoRa-AT andando a considerare la dimensione e la frequenza dei pacchetti inviati. Invece di tentare di equalizzare il Time-on-Air, calcola il numero di simboli trasmessi da ogni ED e modella la capacità dei vari canali in base a quello. Il numero di simboli inviati da un nodo in un certo periodo è dato da:

$$Nsym\_usr_i = Nsym\_mess_i \cdot Nmess_i \tag{3.1}$$

Dove $Nmess_i$ è il numero di pacchetti inviati in un periodo T e $Nsym\_mess_i$ dipende dalla dimensione payload del messaggio (PL) e dal SF usato ed è uguale a:

$$Nsym\_mess = 8 + max\left\{\left\lceil \frac{8PL - 4SF + 28 + 16CR - 20H}{4(SF - 2DE)} \right\rceil \cdot (CR + 4), 0\right\} \tag{3.2}$$

La capacità dei vari canali è equalizzata attraverso la stessa procedura "*ordered water-filling*" usata per EXPLoRa-AT. L'assegnazione degli SF, partendo da quello più piccolo, avviene scorrendo i dispositivi in ordine decrescente rispetto al numero di simboli trasmessi. Inoltre, deve essere sempre verificato che lo SF assegnato sia sufficiente per una corretta comunicazione tra ED e GW più vicino.

3.2 Dispositivi privi di batteria

Bilioni di dispositivi IoT dipendono dalle batterie come fonte di alimentazione principale [10]. Le batterie hanno una vita limitata e non sempre è possibile sostituirle facilmente a causa dei costi, della numerosità e della posizione poco accessibile dei dispositivi. Inoltre sono voluminose, sensibili alla temperatura e non sono in linea con l'attenzione alla sostenibilità che sta nascendo nel mondo IoT [10]. Esistono tecnologie alternative alla batterie che sfruttano fonti energetiche rinnovabili come il sole, il vento e le vibrazioni per alimentare i dispositivi. L'uso di energie rinnovabili presenta ancora svantaggi e sfide che devono essere affrontati, per esempio l'energia raccolta è stocastica e non più deterministica.
Alcune di queste tecnologie sono già in commercio, tra cui interruttori wireless che si ricaricano attraverso l'energia cinetica, smartwatch che raccolgono energia solare e cinetica, sensori collocati su un ponte che sfruttano le vibrazioni prodotte della macchine per caricarsi [11].
La prossima sezione descrive il modello di un dispositivo privo di batteria usato in questa tesi.

3.2.1 Modello di un dispositivo privo di batteria

Per modellare il dispositivo privo di batteria usiamo l'approccio usato in [12]. Il dispositivo è composto da un MCU (Micro Controller Unit), un'unità radio, alcune periferiche, una capacità e un meccanismo per raccogliere l'energia (es. pannello solare). Si può descrivere in modo compatto con un circuito elettrico equivalente diviso in tre parti: (i) harvester, (ii) capacità e (iii) carico.

(i) *Harvester:* unica fonte di energia; può essere modelata come una sorgente di tensione costante ideale (E) con una serie di resistenze che possono variare nel tempo ($r_i(t)$) per modellare la variabilià nel tempo della potenza assorbita. La massima potenza che può essere prodotta è data da:

$$P_h(t) = \frac{E^2}{r_i(t)}. \tag{3.3}$$

Altrimenti, nel simulatore è possibile usare tracce di potenza realistiche ottenute da misure con pannelli solari reali.

(ii) *Carico:* è rappresentato come una resistenza R_L, il suo valore cambia in base allo stato (es. idle, trasmissione) del dispositivo, e modella tutti i componenti del dispositivo che consumano energia.

(iii) *Condensatore:* conserva l'energia raccolta e la rilascia al carico quando richiesta. Per poter funzionare, i dispositivi reali necessitano di una tensione minima V_{th_low}; quando la tensione scende sotto quella soglia il dispositivo si spegne. Si riaccenderà solo quando la capacità si è ricaricata sufficientemente da superare la tensione di soglia $V_{th\_high} > V_{th\_low}$.

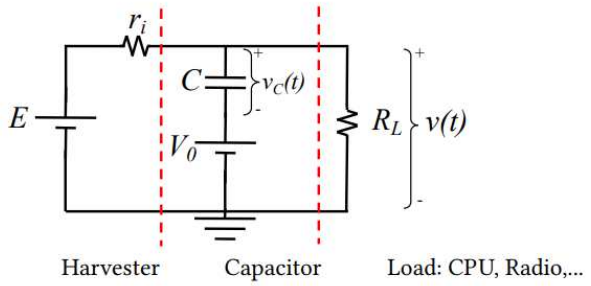

Figura 3.1: Circuito elettrico che modella un dispositivo privo di batteria [12]

Capitolo 4

Simulazione e Analisi

4.1 Il simulatore

Il simulatore di rete usato per questa tesi è ns-3 [13]. È un software open-source che simula eventi discreti ed è scritto nel linguaggio di programmazione C++. Il simulatore tiene traccia di una serie di eventi (per esempio la ricezione o l'invio di un pacchetto) la cui esecuzione è pianificata ad uno specifico tempo di simulazione. Questi eventi saranno poi eseguiti in modo sequenziale senza attendere il reale intervallo di tempo pianificato tra di loro. Questo permette di simulare in pochi minuti quello che in realtà sarebbe stato eseguito in qualche giorno.
Visto che un evento può a sua volta generare un altro evento, la lista dei processi da eseguire potrebbe non svuotarsi mai, per questo è impostata una durata massima dell'esecuzione.

4.2 Modulo `lorawan`

Lo scopo di questa tesi è implementare, tramite il simulatore ns-3, un algoritmo di Adaptive Data Rate per una rete LoRaWAN di dispositivi privi di batteria. È stato possibile farlo grazie alla presenza del modulo `lorawan` che simula i comportamenti di una rete LoRaWAN [14]. Il modulo si occupa della gestione del canale, delle perdite, dell'invio e ricezione dei pacchetti fino al consumo energetico dei dispositivi, fornendo una descrizione della rete e delle sue dinamiche completa e dettagliata.
Di seguito è presente l'analisi solo di alcune classi utili per comprendere il funzionamento degli algoritmi implementati.

- `EndDeviceStatus:` rappresenta le conoscenze del Network Server a proposito degli End Device. Contiene le informazioni sui pacchetti ricevuti: il pacchetto, lo SF con cui è stato trasmesso, la potenza con cui è stato ricevuto dal GW, il tempo di arrivo, il GW che lo ha ricevuto. Inoltre permette di comporre la risposta da mandare all' ED.

- `NetworkStatus:` contiene le conoscenze sullo stato della rete disponibili al Network Server, in particolare le mappe dello stato dei devices con il loro indirizzo e dei Gateways.

- `EndDeviceLorawanMac:` caratterizza la trasmissione a livello MAC. Permette di vedere e modificare i parametri della comunicazione tra cui il *Data Rate Adaptation*, il *Coding Rate* e l'*Header Disabled*.

- `EnergyAwareSenderHelper:` si occupa dell'installazione dell'applicazione EnergyAwareSender sui nodi. Questa applicazione adotta restrizioni sull'invio dei pacchetti in caso di carenza di energia. L'obiettivo che si pone è di evitare il continuo spegnimento e accessione del dispositivo perché porta a un peggioramento delle comunicazioni della rete [5]. Per farlo implementa una verifica a livello MAC che posticipa l'invio dei pacchetti generati a livello applicazione nel caso in cui l'energia disponibile non sia sufficiente alla completa trasmissione del pacchetto e ad evitare lo spegnimento del dispositivo.

- `CapacitorEnergySource:` è la classe principale che si occupa di monitorare e registrare i cambiamenti dell'energia. Rappresenta il condensatore del dispositivo; viene aggiornato il valore dell'energia contenuto attraverso `UpdateEnergySource` e verifica che la tensione non sia minore della soglia V_{th_low} chiamando il metodo `isDepleted`.

- `EndDeviceLorawanMac` e `LoraRadioEnergyModel:` contengono i metodi per il calcolo dei consumi, ad ogni stato in cui si trova il dispositivo corrisponde una corrente differente (specificata nella tabella 4.1) e quindi un consumo energetico diverso. Per calcolare la VoltageThresholdForSuccessfulCycle è necessario considerare i consumi singoli di ogni stato nel tempo trascorso in quello stato.

- `VariableEnergyHarvester:` permette di acquisire da un file esterno i valori di potenza per alimentare i nodi. La classe è stata implementata per leggere file .csv ma

può essere modificata facilmente per leggere file differenti visto che richiede solo la coppia di valori timestamp e P_h. I nodi non assorbono tutti la stessa potenza: essa infatti potrebbe dipendere dalla loro posizione o dalla loro esposizione in base alla sorgente energetica usata. Per questo ad ogni nodo viene conferito un valore di potenza dato da una variabile aleatoria Gaussiana con media pari alla potenza indicata nella traccia in input e di varianza pari al valore indicato in fase di simulazione.

In questo studio sono state usate due tracce contenenti i dati raccolti da un pannello solare in una giornata di sole e in una giornata nuvolosa di Settembre dalle ore 10:00 alle 19:00. Sono state utilizzate per un'analisi realistica del funzionamento della rete in un caso sfavorevole e favorevole.

State	**MCU**	**Radio Current**	**Total current**
Off	Standby	0	5.5μ A
Turn On	Active	-	15mA
Sleep	Active	1μ A	5.6μ A
Tx	Active	28mA	28.011mA
Idle	Standby	1.5μ A	7μ A
Standby	Standby	10.5mA	10.5055mA
Rx	Active	11mA	11.011mA

Tabella 4.1: Correnti associate allo stato del dispositivo

Il paragrafo succesivo descrive il funzionamento degli algoritmi implementati in questa tesi.

4.3 Contributo

Gli algoritmi ADR vengono eseguiti lato NS, vengono chiamati ogni qualvolta viene ricevuto un pacchetto ed eseguiti solo se quel pacchetto lo richiede. Contengono come parametri in ingresso sia l'`EndDeviceStatus` che il `NetworkStatus`.

4.3.1 `AdrLoad`

AdrLoad è la classe che implementa l'algoritmo *EXPLoRa-TS*. Questo algoritmo bilancia il traffico della rete (misurato in tempi di simbolo) tra i vari SF. Possiamo immaginare i canali come contenitori dove i simboli associati agli ED vengono opportunamente inseriti

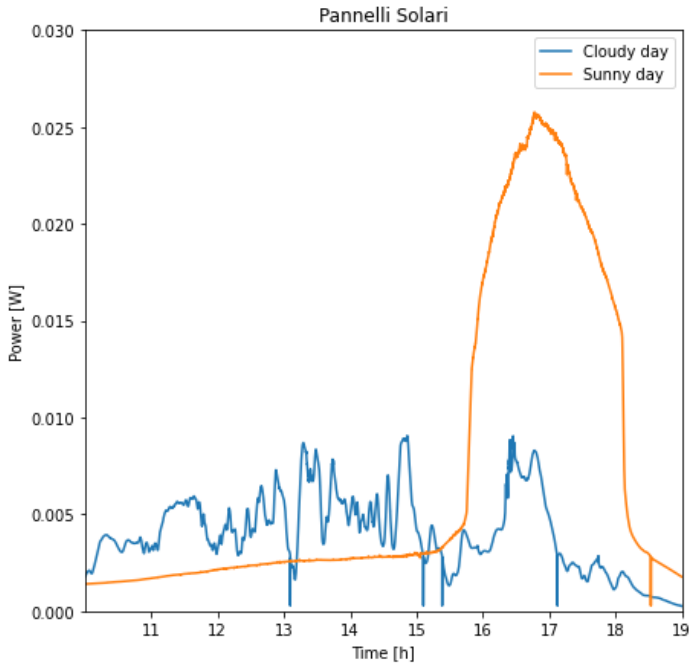

Figura 4.1: Tracce di potenza assorbita dai pannelli solari

fino a riempimento (Fig. 4.2).

Il numero di simboli per messaggio dipende dalla dimensione del payload (PL) in bytes,

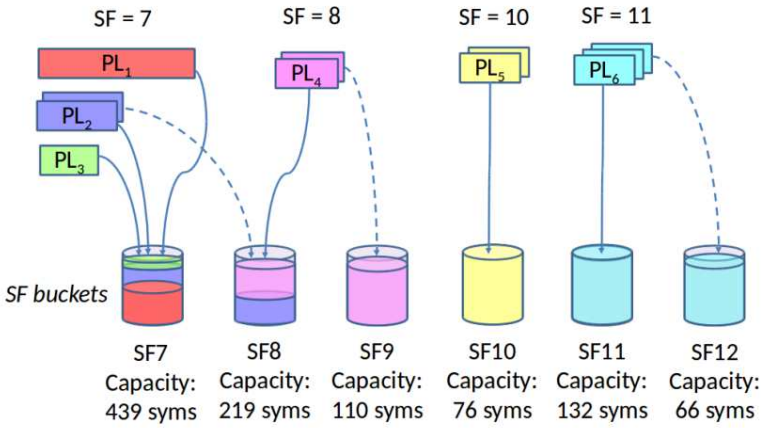

Figura 4.2: Esempio di EXP_*water_filling* [7]

dallo SF, dal Code Rate (CR), dall'header (H) e dall'ottimizzazione del Data Rate (DE). Viene calcolato come segue:

$$Nsym\_mess = 8 + max\left\{\left\lceil \frac{8PL - 4SF + 28 + 16CR - 20H}{4(SF - 2DE)} \right\rceil \cdot (CR + 4), 0\right\} \quad (4.1)$$

Come illustrato in Alg.1, la procedura richiede come ingresso i vettori PL_{vec}, MP_{vec}, SF_{vec}, CR_{vec}, DE_{vec}, H_{vec}, i primi tre sono rispettivamente dimensione payload, il tempo tra l'invio di un messaggio e il successivo e SF. Il calcolo del numero di simboli per ED ($Nsym\_usr_i$) viene eseguito come:

$$Nsym\_usr_i = Nsym\_mess_i \cdot Nmess_i. \tag{4.2}$$

Infine P_{ADR} contiene il carico del traffico della rete pesato in base allo SF a cui si riferisce; verrà prima equalizzato e successivamente usato per definire la capacità di ogni canale (B).

Algorithm 1 EXPLoRa-TS

function EXPLORA-TS($PL_{vec}, MP_{vec}, SF_{vec}, CR_{vec}, DE_{vec}, H_{vec}$)
    $w \leftarrow 1, 2, 3, 4, 8, 16, 32$
    $Nmess_i \leftarrow 1/MP_i \ \forall i \in [1...N]$
    $Nsym\_mess_i \leftarrow PLsym(PL_i, SF_i, CR_i, H_i, DE_i)$
    $Nsym\_usr_i \leftarrow Nsym\_mess_i \cdot Nmess_i$
    $Nsym\_SF_i \leftarrow$ *numero di simboli per* $SF_k \ \forall k \in [7...12]$
    $P_{ADR} \leftarrow Nsym\_SF \cdot w$
    $P_{TS} \leftarrow EXP\_water\_filling(P_{ADR}, w)$
    $B \leftarrow P \cdot w^{-1}$
    $S \leftarrow fill\_buckets(B)$
return S
end function

La seconda parte della soluzione si occupa di distribuire i messaggi nei vari contenitori. Nel farlo abbiamo due limitazioni: (i) il numero di simboli per messaggio cambia in base al contenitore scelto; (ii) ogni ED ha uno SF = k minimo, non è possibile scegliere k-1 perché il GW non sarebbe in grado di decifrare il messaggio a causa della soglia di sensibilità associata a quello SF. La soluzione proposta (Alg. 2) inizia allocando i messaggi con PL maggiore per aumentare le probabilità che lo spazio rimasto sia sufficiente per gli ED con PL di taglie minori. Ogni volta che viene inserito un messaggio in un contenitore la sua capacità rimanente decresce e viene associato all'ED lo SF corrispondente. Infine verrà comunicato ad ogni ED, tramite un messaggio ADR, il nuovo SF da usare per trasmettere i pacchetti, se questo è stato modificato.

Algorithm 2 DataRateAllocation

```
function DataRateAllocation(B, Address, SF, Nmess, Nsym_mess, CR, H, DE)
    t ← numero di dispositivi
    for i ← 0 to 5 do
        SymToAllocate ← B_i
        while SymToAllocate > 0 AND enRoom AND t > 0 do
            max ← indice ED con PL maggiore
            if i! = SF_max then
                Nsym_mess_max ← PLsym(PL_max, i + 7, CR_max, H_max, DE_max)
            end if
            if SymToAllocate > Nsym_mess_max · Nmess_max then
                *Controllare se nuovo SF maggiore del minimo*
                map ← insert(Address_max, newDR)
                t = t-1
            else
                for j ← 0 to numDispositiviNoti do
                    if SymToAllocate >= Nsym_mess_j · Nmess_j then
                        map ← insert(Address_j, newDR)
                        SymToAllocate ← SymToAllocate − Nsym_mess_j · Nmess_j
                        t = t-1
                    end if
                end for
                enRoom ← false
            end if
        end while
        if i < 5 AND SymToAllocate > 0 then
            B_{i+1} ← +SymToAllocate
        end if
    end for
return map
end function
```

4.3.2 AdrLoadEnergyHarvesting

EXPLoRa-TS è un ottimo algoritmo di ADR ma è stato pensato per dispositivi alimentati a batteria. L'obiettivo di questa sezione è di proporre una serie di modifiche di AdrLoad in modo da ottenere degli algoritmi pensati per reti di dispositivi privi di batteria.

AdrLoadEnergyHarvestingA

Ricordando dalla sezione 2.1.1 che SF minori richiedono meno energia per trasmettere un pacchetto a parità di dimensione, in questa soluzione è stato scelto di prioritizzare i dispositivi meno carichi e non più quelli con PL maggiore.
La struttura principale dell'algoritmo è analoga ad AdrLoad, le modifiche effettuate sono state:

(a) Raccogliere i dati riguardanti la tensione ai capi del condensatore di ogni ED (`ActualVoltage`) e aggiungerli ai parametri di Alg. 2.

(b) Invece che cercare l'indice del dispositivo con PL massimo, ricavare l'indice del dispositivo con il valore di tensione minimo. Quindi partire da quello per l'assegnazione degli SF.

AdrLoadEnergyHarvestingB

AdrLoadEnergyHarvestingB è una variante più complessa di AdrLoadEnergyHarvestingA. In questo caso, l'obiettivo è tenere conto sia della dimensione del pacchetto, sia del livello di carica del dispositivo. Per fare ciò salva i dati riguardanti la tensione ai capi del condensatore e la dimensione del PL dei messaggi inviati da ogni ED. Successivamente calcola l'energia in relazione alla dimensione del pacchetto da inviare, cioè calcola il rapporto tra `ActualVoltage/PL`.
Questo valore sarà usato per scegliere l'ordine con cui assegnare gli SF ai dispositivi; i dispositivi con un rapporto minore avranno la precedenza sull'assegnazione di SF minori. In pratica invece di ricercare l'indice associato all'ED con PL massimo, questo algoritmo ricerca l'indice associato all'ED con rapporto minore ed inizia l'assegnazione con quello.

AdrLoadEnergyHarvestingC

L'ultima versione implementata è AdrLoadEnergyHarvestingC. Questa soluzione sfrutta il concetto di `VoltageThresholdForSuccessfulCycle`: tensione minima necessaria per

permettere al dispositivo di completare la trasmissione di un pacchetto noto, usando lo SF indicato, senza scendere sotto la tensione di soglia V_{th_low}.
Il calcolo di questa soglia avviene in modo conservativo, si riferisce alla tensione necessaria nel caso l'energia dall'ambiente durante la trasmissione sia nulla. L'algoritmo non modifica l'ordine di assegnazione definito da EXPLoRa-TS ma va a verificare che la scelta fatta sia compatibile con l'energia del dispositivo. Se così non fosse prova a verificare se l'invio dello stesso pacchetto con la stessa energia andrebbe a buon fine usando uno SF minore, in caso positivo assegna quello. Non sempre è possibile assegnare uno SF minore a causa delle limitazioni sulla distanza raggiungibile che esso produce, in quel caso verrà scelto il minor valore accettabile.

4.4 Parametri di simulazione

L'esecuzione di una simulazione si può suddividere in diversi passaggi, quelli di maggior interesse per il nostro studio sono:

1. **Creazione del canale:** definisce il path loss dovuto al canale, è dato da due contributi: *propagation loss*, che dipende dalla distanza tra emettitore e ricevitore; *building penetration loss*, dovuto all'attenuazione dei muri degli edifici presenti [15]. Gli edifici sono creati e posizionati in una griglia 2D dalla classe `GridBuildingAllocator`.

2. **Creazione ed installazione ED:** i nodi vengono creati, tanti quanta la dimensione della rete richiesta, gli viene assegnata una posizione, un indirizzo, una tipologia e viene definito il tipo di comunicazione. In questo caso di studio viene sempre usata una comunicazione non confermata, cioè non è richiesta la conferma di ricezione da parte del NS. La distribuzione degli ED avviene in modo uniforme all'interno di un'area circolare di raggio pari a 2500m con il GW posto al centro.

3. **Creazione GW**.

4. **Installazione applicazioni ED:** definisce le modalità di invio dei pacchetti per ogni ED. Il tempo tra un invio e il successivo e la dimensione del pacchetto inviata sono i valori dati da due variabili aleatorie uniformi non nulle negli intervalli [600,1800] secondi e [10,40] bytes. Se non avessimo scelto ED con specifiche di invio dei pacchetti diverse tra loro, verrebbe assegnato a tutti il valore medio delle due variabili aleatorie.

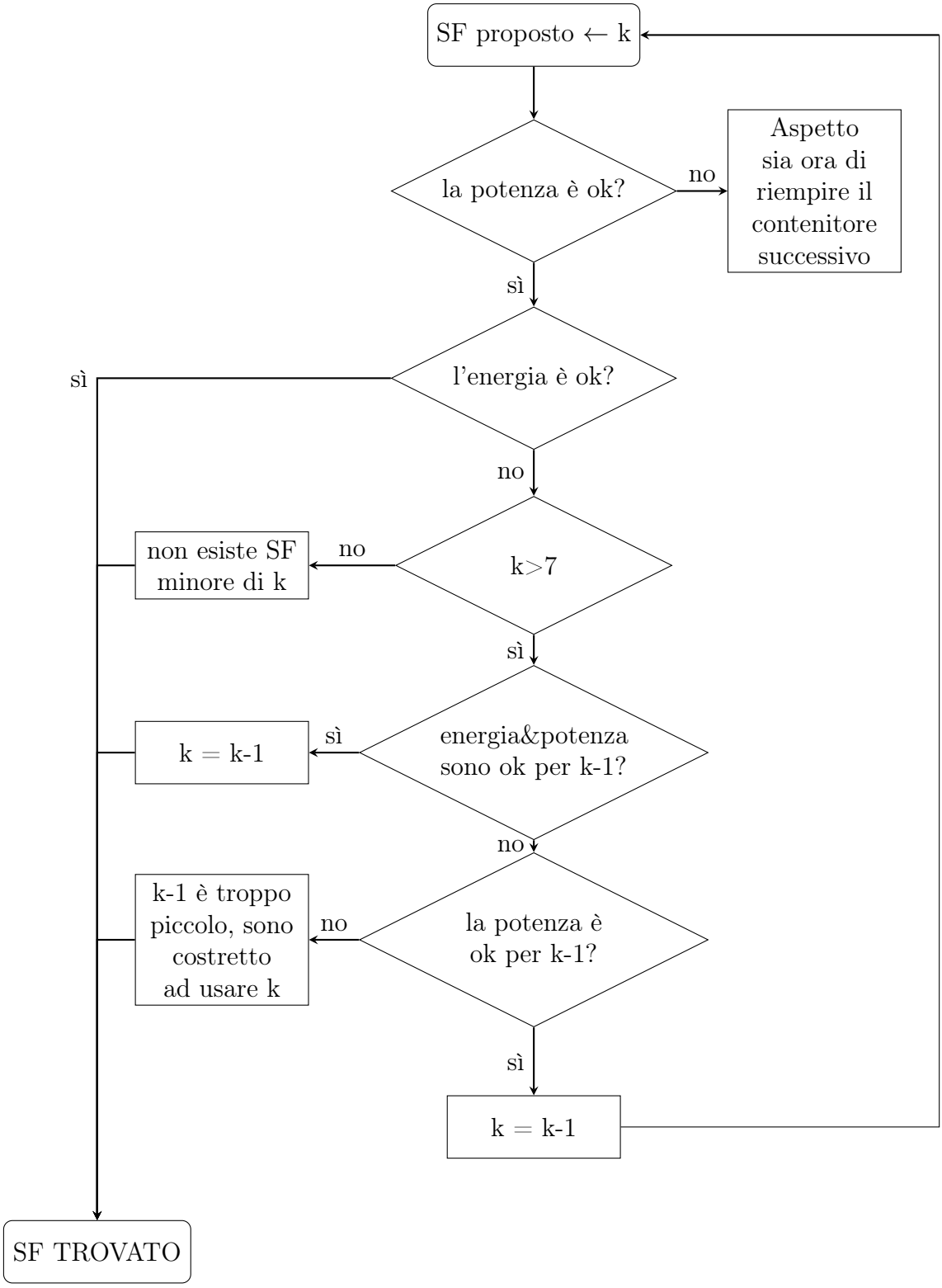

Figura 4.3: Procedura di asseganzione degli SF in AdrLoadEnergyHarvestingC

5. **Definizione del modello energetico:** imposta la capacità del condensatore pari a 40mF, V_{th_low}=1.8V, V_{th_hight}=3V e V_{max}=3.3V. La potenza non è definita costante tra tutti i dispositivi ma come il valore di una variabile aleatoria Gaussiana di media pari a *eh* e varianza *maxVariance*, entrambi questi parametri sono modificabili per le analisi.

6. **Installazione Network Server:** unisce gli elementi che sono stati creati precedentemente come EDs e GWs; imposta quale algoritmo di ADR eseguire.

7. **Esecuzione della simulazione:** una volta che la modellazione della rete viene terminata è possibile eseguire la simulazione per la durata desiderata, in questo studio è stata scelta pari a 3600s.

8. **Tracciamento dati:** durante l'esecuzione, ns-3 permette di tenere traccia di eventi e cambiamenti di stato della rete. Nel presente caso rileviamo i pacchetti inviati e ricevuti a livello MAC, i pacchetti generati a livello applicazione, il numero di finestre RX_{1-2} aperte e mancate, il numero di pacchetti ricevuti da ogni GW.

9. **Analisi dati:** in secondo luogo avviene l'analisi dei dati estrapolati, un modulo Python calcola il Data Extraction Rate (DER) che è dato dai pacchetti ricevuti fratto i pacchetti generati a livello Applicazione. Il DER viene visualizzato in funzione delle dimensioni della rete per capire come l'algoritmo di ADR usato si comporti scalando la rete.

Capitolo 5

Analisi dei risultati

5.1 Confronto tra le versioni implementate

Le versioni di AdrLoadEnergyHarvesting si pongono l'obiettivo di migliorare le prestazioni di AdrLoad in una rete di dispositivi privi di batteria. Per raggiungere questo scopo essi sfruttano diversi metodi. Attraverso i risultati ottenuti possiamo determinare quale di queste strategie è la migliore.
Il confronto tra le tre versioni di AdrLoadEnergyHarvesting implementate è stato effettuato mantenendo gli altri parametri costanti: in particolare, i dispositivi sono alimentati con una potenza costante pari a 0.01W e una deviazione stardard pari a 0.1W. In tutte le simulazioni, se non specificato diversamente, dimensione dei pacchetti e frequenza di invio sono diversi per ogni nodo della rete.

5.1.1 AdrLoadEnergyHarvestingA

Come si può vedere dalla Fig. 5.1 il nuovo algoritmo non migliora le prestazioni di AdrLoad raggiungendo risultati simili. Nonostante la carica di un dispositivo sia un fattore importante da considerare per evitare il suo spegnimento, non è prioritario. Il consumo energetico è fortemente influenzato dallo SF e dalla dimensione del pacchetto inviato; l'invio di pacchetti di dimensioni elevate tramite uno SF alto è difficilmente sostenibile anche per dispositivi carichi.

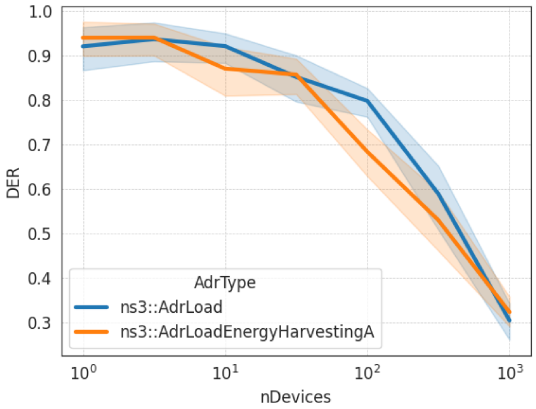

Figura 5.1: Confronto prestazioni tra AdrLoad e AdrLoadEnergyHarvestingA

5.1.2 AdrLoadEnergyHarvestingB

I risultati ottenuti precedentemente (sez. 5.1.1) mostrano come non ci si può limitare a considerare solo la carica dei dispositivi.
Questa seconda soluzione considera quindi il livello di carica in relazione alla dimensione del pacchetto per assegnare gli SF. Infatti, ED carichi ma che spediscono pacchetti di dimensione elevata devono avere la precedenza nell'assegnazione di SF minori rispetto a dispositivi meno carichi i cui messaggi hanno una dimensione ridotta (vedi sez. 4.3.2).
I risultati ottenuti non presentano miglioramenti rispetto ad AdrLoad, né particolari miglioramenti rispetto ad AdrLoadEnergyHarvestingA.
Per questa motivazione la strategia proposta da AdrLoadEnergyHarvestingC si discosta completamente da quelle fino ad ora discusse.

5.1.3 AdrLoadEnergyHarvestingC

AdrLoadEnergyHarvestingC non modifica l'ordine di assegnazione degli SF rispetto ad AdrLoad, essa agisce sullo SF inizialmente assegnato solo quando questo non permette la comunicazione completa del messaggio perché richiede una tensione maggiore di quella presente ai capi del condensatore: in quel caso viene assegnato uno SF minore.
Questo potrebbe discostare il carico di alcuni canali rispetto alla loro capacità ottima come definita da *EXPLoRa-TS*, ma i vantaggi dati dall'efficienza energetica di questo metodo superano il possibile lieve incremento delle collisioni.
Confrontando Fig. 5.3 con quelle precedenti si può notare il miglioramento di prestazioni

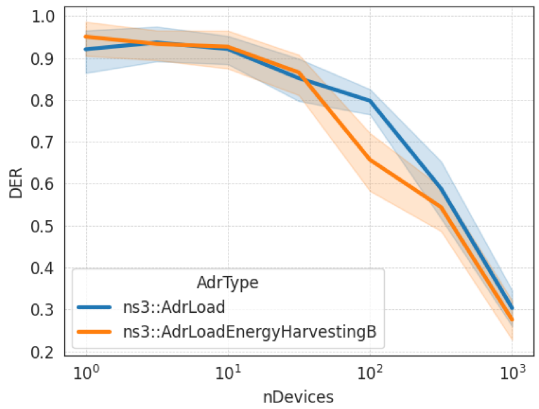

Figura 5.2: Confronto prestazioni tra AdrLoad e AdrLoadEnergyHarvestingB

raggiunto da AdrLoadEnergyHarvestingC rispetto alle altre versioni.
Rispetto ad AdrLoad l'andamento del DER è equivalente in caso di reti di dimensioni limitate, fino a qualche decina di dispositivi. All'aumentare della dimensione si può notare l'incremento apportato da AdrLoadEnergyHarvestingC.
Dati gli incoraggianti risultati ottenuti da questo algoritmo, nelle prossime analisi sarà l'unico ad essere considerato e prenderà il nome di AdrLoadEnergyHarvesting.

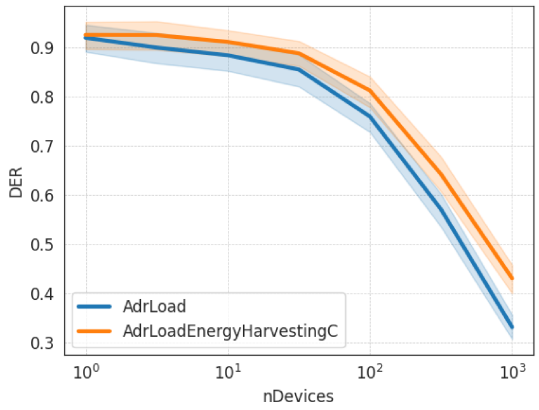

Figura 5.3: Confronto prestazioni tra AdrLoad e AdrLoadEnergyHarvestingC

5.2 Prestazioni della rete con diverse fonti energetiche

Anche se il risultato ottenuto da AdrLoadEnergyHarvesting è promettente, lo scenario con cui è stato ottenuto, cioè una potenza in ingresso costante, è poco realistico. In questo paragrafo vengo confrontate le prestazioni di AdrLoad e AdrLoadEnergyHarvesting in due differenti scenari energetici verosimili. In entrambi gli scenari è considerata la media dei risultati ottenuti in caso di deviazione standard di potenza pari a 0.1W, 0.07W, 0.03W.

5.2.1 Giornata soleggiata

In questa analisi è stata utilizzata una traccia reale della potenza assorbita da un pannello solare in una giornata soleggiata di settembre. La traccia copre l'intervallo di tempo tra le 10 di mattina fino alle 19 (Fig. 4.1). Il grafico in Fig. 5.4 mostra come il nuovo algoritmo permetta di migliorare il valore del DER fino a un fattore pari a 4%. È presente un miglioramento anche all'aumentare della dimensione della rete.

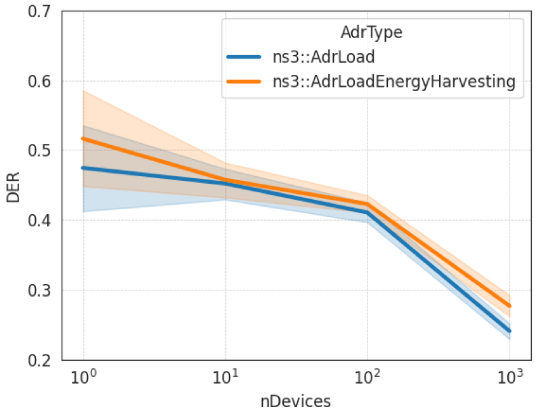

Figura 5.4: Confronto prestazioni tra AdrLoad e AdrLoadEnergyHarvesting in una giornata di sole

5.2.2 Giornata nuvolosa

In questo caso è stata utilizzata una traccia reale della potenza assorbita da un pannello solare in una giornata nuvolosa di settembre. I dati sono stati raccolti tra le 10 di mattina fino alle 19 (Fig. 4.1). Il grafico in Fig. 5.4 mostra la presenza di un miglioramento fino a un fattore pari al 5%.
Si può notare come in generale si ottengono prestazioni migliori nel caso della giornata

nuvolosa. Il valore del DER nel primo caso parte da 0.51 per poi decrescere fino a 0.28 col crescere della rete, nel secondo caso il valore massimo si aggira attorno allo 0.56 per poi scendere fino allo 0.30 circa. Questo risultato potrebbe risultare controintuitivo. Guardando le tracce in Fig. 4.1 con attenzione, si può notare che durante la giornata nuvolosa viene raccolta una energia relativamente costante per la maggior parte della giornata. Durante la giornata di sole l'energia raccolta per la prima metà della giornata è inferiore rispetto alla traccia precedente, durante il pomeriggio è presente un picco. Questo spiega come, mediamente, la giornata nuvolosa permetta di spedire più pacchetti correttamente rispetto all'altro scenario.
I miglioramenti ottenuti dall'utilizzo di AdrLoadEnergyHarvesting si mantengono anche all'aumentare della dimensione della rete.

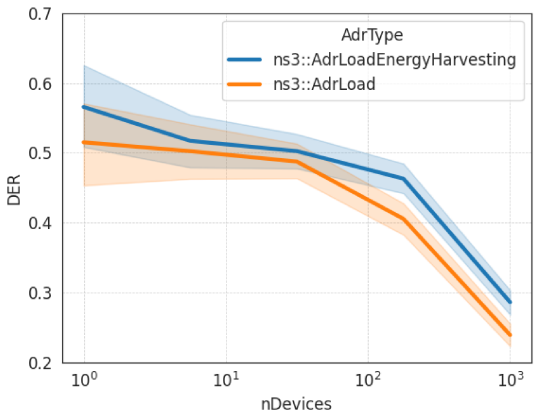

Figura 5.5: Confronto prestazioni tra AdrLoad e AdrLoadEnergyHarvesting in una giornata di pioggia

5.3 Prestazioni della rete in funzione della varianza di potenza

Le analisi effettuate simulano una rete di dispositivi distribuita all'interno di una città. La presenza di edifici e altre strutture tra il sole e il dispositivo o la diversa esposizione portano a potenze assorbite associate ad ogni nodo molto differenti tra loro a parità di situazioni meteorologiche. Non essendo stato possibile trovare delle tracce sperimentali che misurassero questo effetto sono state usate delle variabile aleatorie con una varianza definita per simularle.
Il grafico in Fig. 5.6 mostra come i diversi valori assegnati alla varianza della potenza

generata dall'EH non modifichino sensibilmente le prestazioni della rete. Questo rende l'algoritmo robusto ai possibili cambiamenti di esposizione dei dispositivi a lungo termine. Per esempio, in uno scenario urbano, la crescita di un albero o la costruzione di un edificio potrebbe oscurare una serie di sensori modificando la varianza dell'intera rete. Inoltre lo rende flessibile: è possibile utilizzarlo sia in applicazioni dove la potenza assorbita dai vari dispositivi è eterogenea, sia quando la potenza assorbita è omogenea.

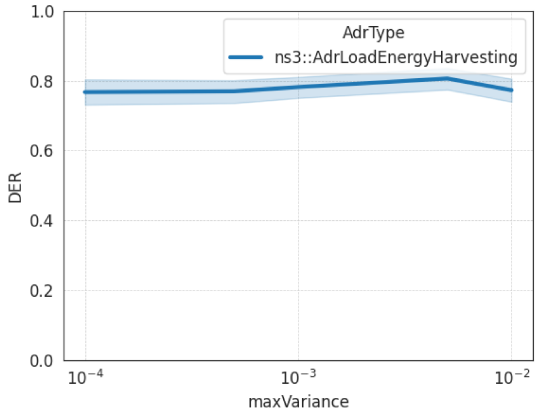

Figura 5.6: Comportamento di AdrLoadEnergyHarvesting al variare della varianza di potenza con potenza media pari a 0.01W

Capitolo 6

Conclusione

In questa tesi è stata presentata l'implementazione di un algoritmo di adaptive data rate per reti LoRaWAN con dispositivi privi di batteria.
I risultati prodotti dalle simulazioni hanno evidenziato come la terza versione del codice implementato è l'unica che ha prodotto i risultati desiderati in termini di miglioramento della DER e per questo l'unica usata per le simulazioni successive. Le simulazioni sono state eseguite utilizzando come ingresso delle tracce empiriche di potenza generata da pannelli solari. I risultati hanno mostrato che questi dispositivi, comunicando attraverso una tecnologia LPWAN, non richiedono molta energia per ottenere buone prestazioni ma ne richiedono un livello minimo costante. Questo aspetto è un fattore da considerare nella scelta del tipo di fonte rinnovabile da sfruttare per alimentare i dispositivi, e va determinato anche in funzione dello scenario.
Le analisi effettuate hanno anche dimostrato che nelle situazioni critiche di applicazione, dove non è sempre presente un'energia minima costante, AdrLoadEnergyHarvesting ottiene dei miglioramenti non trascurabili rispetto ad AdrLoad.

I risultati preliminari descritti in questa analisi possono essere ulteriormente approfonditi per confermare le conclusioni tratte da questa tesi. In particolare è necessario un data set di tracce di potenza più vasto che rappresenti meglio gli scenari di applicazione di questo algoritmo. Data questa premessa i risultati ottenuti rimangono incoraggianti.

Bibliografia

[1] M. P. M. Chui M. Collins. "IoT value set to accelerate through 2030: Where and how to capture it." (), indirizzo: https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/iot-value-set-to-accelerate-through-2030-where-and-how-to-capture-it.

[2] L. Vangelista, "Frequency Shift Chirp Modulation: The LoRa Modulation," *IEEE Signal Processing Letters*, vol. 24, n. 12, pp. 1818–1821, 2017. DOI: 10.1109/LSP.2017.2762960.

[3] F. Cuomo, J. C. C. Gámez, A. Maurizio et al., "Towards traffic-oriented spreading factor allocations in LoRaWAN systems," in *2018 17th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*, 2018, pp. 1–8. DOI: 10.23919/MedHocNet.2018.8407091.

[4] L. Alliance. "LoRaWANTM 1.1 Specification." (), indirizzo: https://resources.lora-alliance.org/technical-specifications/lorawan-specification-v1-1.

[5] M. Capuzzo, C. Delgado, A. Sultania, J. Famaey e A. Zanella, "Enabling Green IoT: Energy-Aware Communication Protocols for Battery-less LoRaWAN Devices," nov. 2021, pp. 95–98. DOI: 10.1145/3479239.3485676.

[6] D. Halperin, M. J. Ammer, T. E. Anderson e D. Wetherall, "Interference Cancellation: Better Receivers for a New Wireless MAC," in *HotNets*, 2007.

[7] F. Cuomo, M. Campo, A. Caponi, G. Bianchi, G. Rossini e P. Pisani, "EXPLoRa: Extending the performance of LoRa by suitable spreading factor allocations," *2017 IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pp. 1–8, 2017.

[8] A. Mahmood, E. Sisinni, L. Guntupalli, R. Rondón, S. A. Hassan e M. Gidlund, "Scalability Analysis of a LoRa Network Under Imperfect Orthogonality," *IEEE Transactions on Industrial Informatics*, vol. 15, pp. 1425–1436, 2019.

[9] P. Kumari, H. P. Gupta e T. Dutta, "An Incentive Mechanism-Based Stackelberg Game for Scheduling of LoRa Spreading Factors," *IEEE Transactions on Network and Service Management*, vol. 17, pp. 2598–2609, 2020.

[10] A. Sabovic, C. Delgado, D. Subotic, B. Jooris, E. D. Poorter e J. Famaey, "Energy-Aware Sensing on Battery-Less LoRaWAN Devices with Energy Harvesting," *Electronics*, 2020.

[11] F. Orfei, C. B. Mezzetti e F. Cottone, "Vibrations powered LoRa sensor: An electromechanical energy harvester working on a real bridge," *2016 IEEE SENSORS*, pp. 1–3, 2016.

[12] M. Capuzzo, C. Delgado, J. Famaey e A. Zanella, "An ns-3 implementation of a battery-less node for energy-harvesting internet of things," giu. 2021, pp. 57–64. DOI: 10.1145/3460797.3460805.

[13] nsnam. "The ns-3 network simulator." (), indirizzo: :https://www.nsnam.org/.

[14] D. Magrin, M. Capuzzo, S. Romagnolo e M. Luvisotto, *LoRaWAN ns-3 module*, https://https://github.com/signetlabdei/lorawan.

[15] D. Magrin, M. Centenaro e L. Vangelista, "Performance evaluation of LoRa networks in a smart city scenario," in *2017 IEEE International Conference on Communications (ICC)*, 2017, pp. 1–7. DOI: 10.1109/ICC.2017.7996384.