Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Informatica Magistrale

STUDIO ED ANALISI DI PRESTAZIONI DI RETI DI UAV BASATE SU TECNOLOGIA LTE SIDE-LINK

Relatore:
Chiar.mo Prof.
MARCO DI FELICE

Presentata da:
FEDERICO DALE'

Correlatori:
Chiar.mo Prof.
ENRICO NATALIZIO
Chiar.mo Prof.
ANGELO TROTTA

Sessione III
Anno Accademico 2019/2020

Ai miei genitori:

per avermi permesso di coltivare la mia più grande passione,

senza mai farmi mancare nulla ...

Introduzione

Alla base di questa tesi vi è lo studio di UAVs (Unmanned Aerial Vehicles) Swarm Network che utilizzano LTE Side-Link per la comunicazione fra nodi. In particolare, si valutano le prestazioni della rete in termini di Throughput, Delay medio e PDR di un sistema simulato con OMNeT++ con numero di UAVs variabile, che come tecnologia di comunicazione utilizza LTE Side-Link Mode 4. Nel sistema simulato l'obbiettivo è quello di consegnare tutti i pacchetti creati dal mittente al destinatario e valutare le prestazioni generali di questo algoritmo e della suddetta tecnologia utilizzando le metriche sopracitate.

Side-Link é progettato per la comunicazione diretta tra due nodi della rete e non implementa alcuna funzione di routing dei messaggi verso UAVs piú lontani. L'obbiettivo di questo studio é aggiungere la comunicazione multihop al Side-Link, in modo tale da poter ovviare alla problematica descritta. Le prestazioni vengono valutate sulla base dell'algoritmo di routing che si occupa di gestire l'inoltro dei pacchetti verso il nodo di destinazione. La tecnologia scelta permette di comunicare senza l'utilizzo di una stazione base, punto fondamentale per l'utilizzo dei device in tutti luoghi, anche remoti.

La tesi è articolata in due parti: nella prima viene fornita un'introduzione allo stato dell'arte del mondo delle FANET e degli UAVs in generale, soffermandosi sulla tecnologia utilizzata nel progetto. Nella seconda ci si occupa di sviluppare un'analisi completa del modello di simulazione. Questa parte è divisa in 3 capitoli: il primo si concentra sulla modellazione dell'intero sistema, descrivendo l'ambiente che si vuole creare ai fini della simulazione e

l'algoritmo proposto. Nel secondo capitolo l'analisi si concentra sull'implementazione. Si cerca di descrivere nei minimi particolari tutte le componenti utilizzate e il codice sorgente dell'algoritmo. Nel terzo, infine, si procede alle vere e proprie valutazioni, a partire dalle varie simulazioni sulla rete in esame.

Grazie a questo lavoro di ricerca è stato possibile analizzare alcuni importanti fattori legati alle reti FANET ed avere un primo approccio al mondo della simulazione e della ricerca in questo ambito. Inoltre è stato possibile valutare e testare questo tipo di tecnologia, utilizzando vari framework e confrontandone pregi e difetti.

Indice

Elenco delle figure

Elenco delle tabelle

Parte I

Stato dell'Arte

Capitolo 1

UAVs

Un UAV, noto comunemente come drone, è un apparecchio volante caratterizzato dall'assenza del pilota a bordo. Il suo volo è controllato via software, dal computer a bordo del mezzo aereo, oppure tramite il controllo remoto di un navigatore o pilota, sul terreno o in altre posizioni. Esistono vari tipi di droni, ognuno con diverse caratteristiche, da quelli progettati per essere controllati da un radio comando ed essere pilotati unicamente per scopi ludici, a quelli più avanzati, con un controller e una memoria a bordo, che possono eseguire software specifici, utili a una causa precisa. In questo capitolo verrà introdotto il concetto di UAV patendo dalla nascita, fino ad arrivare alle applicazioni moderne, sia in ambito civile che in quello militare.

1.1 Breve Storia

Il primo tentativo di costruire e utilizzare un UAV risale al 1849, quando gli Austriaci attaccarono la città di Venezia utilizzando dei palloni carichi di esplosivo, lanciati dalla nave Vulcano. Sebbene alcuni dei palloni avessero funzionato, altri, a causa del vento, finirono per colpire le linee di attacco alleate.

I successivi esempi e prototipi di velivoli senza pilota fecero la loro comparsa durante la prima guerra mondiale: il primo esempio è l'Aerial Target. Sempre durate lo stesso conflitto, l'aeroplano automatico Hewitt – Sperry, o anche noto come bomba volante, compì il suo primo volo, dimostrando il concetto di aereo senza pilota. Il velivolo veniva comandato grazie ad una serie di giroscopi montati internamente. Nel periodo di tempo compreso tra le guerre mondiali, l'appassionato di aeromobili americano Reginald Denny creò alcuni modelli radio-controllati a basso costo denominati RP-1. Questi vennero utilizzati sia durante il secondo conflitto mondiale, sia successivamente. Durante la Guerra Fredda e del Vietnam, lo sviluppo tecnologico permise di raggiungere un elevato livello qualitativo, portando sul mercato soluzioni sempre più piccole e con caratteristiche tali da poter essere impiegati in innumerevoli scenari operativi.

La ricerca e lo sviluppo di questo tipo di sistemi è sempre avvenuto in ambito militare e tutte le prime sperimentazioni sono state fatte a seguito di determinate esigenze dovute a questa pratica. Negli anni 2000, grazie al rapido progresso tecnologico, i droni hanno fatto la loro comparsa anche in ambito civile, dove sono stati impiegati inizialmente nella sorveglianza delle aeree di coltivazione e in aerofotogrammetria, e poi per effettuare riprese aeree cinematografiche, in operazioni di ricerca e salvataggio, nel controllo di linee elettriche e condutture petrolifere e nel monitoraggio della fauna selvatica. In questo ambito, nei paesi anglosassoni, viene utilizzato in prevalenza il termine "drone" piuttosto che quello originario UAV ("unmanned aerial vehicle").

1.2 Utilizzi in ambito civile

Negli ultimi anni, le tecnologie utilizzate per la costruzione degli UAVs sono migliorate, sopratutto la tecnologia sensoristica, che permette di equipaggiare un drone con un qualsiasi tipo di sensore, date le piccole dimensioni di quest'ultimo. Infatti, molte aziende hanno prodotto esemplari con molteplici rilevatori, che variano da quelli ottici, come ad esempio una videocamera utile per fare riprese aeree, fino ad arrivare ai termici oppure Lidar, che permettono di misurare la distanza da un oggetto. Un esempio é la casa produttrice "DJI", leader del commercio di droni radiocomandati, che propone diversi modelli che implementano un giroscopio, un sensore di prossimità per evitare gli ostacoli e per automatizzare l'atterraggio e un dispositivo GPS per la localizzazione.

L'ambito applicativo però è molto diverso da quello teorico, infatti l'utilizzo di ogni tipo di drone è subordinato alla regolamentazione in vigore nel paese in cui deve volare, dove per esempio vengono limitate le distanze tra i dispositivi e il centro di controllo. Per esempio, in Europa é vietato il volo al disopra di grandi folle e la regolamentazione é molto dettagliata in quanto a peso e certificazioni del prodotto utilizzato. Alcuni scenari applicativi possono essere:

- **Sicurezza territoriale, delle frontiere e lotta ai narcotrafficanti**
- **Monitoraggio siti Archeologici, contro la depredazione e il commercio illegale di reperti**
- **Monitoraggio centrali termoelettriche e impianti industriali**
- **Telerilevamento**
- **Aerofotogrammetria e rilievo dell'architettura**
- **Monitoraggio ambientale e calamità naturali**
- **Biodiversità e monitoraggio fauna**

- **Operazioni di ricerca e soccorso**

- **Videoriprese e fotografie**

Più in generale, i droni sono utili in tutte quelle missioni caratterizzate da un alto rischio per la vita umana, oppure per raggiungere aree inaccessibili o impervie. In tutti questi scenari, la presenza di una videocamera sul UAV è d'obbligo e grazie a questo tipo di riprese si riesce a portare a termine il compito finale.

Ognuno di questi scenari prevede inoltre un territorio da monitorare, di solito abbastanza vasto, perciò un UAV soltanto non riuscirebbe a tenere tutto sotto controllo. Per offrire un servizio su vasta scala è necessario utilizzare più UAVs che comunichino tra di loro per arrivare ad uno scopo finale comune. Per esempio, per monitorare un territorio coltivato di diversi chilometri quadrati sono necessari piú droni che collaborano, ognuno sorvolando una porzione dell'area. Da qui nasce quindi il bisogno di collegare ogni dispositivo a tutti gli altri, mettendoli in rete e garantendo una comunicazione costante fra loro e con la stazione di terra, che ha il compito di raccogliere tutti i dati creati nella rete. Solo così si può raggiungere un risultato comune e completare la missione richiesta dallo scenario.

1.3 UAVs Networks

Una rete di UAVs, classificata anche come swarm network, è un insieme di droni che costituiscono i vari nodi che comunicano tra di loro e con la stazione di terra (Ground Station). Questa rete deve essere veloce e sicura, inoltre deve permettere lo scambio rapido di messaggi tra i vari componenti, in tempo reale e in modo tale da garantire continuità nella comunicazione. Un esempio di questa tipologia di reti applicata al mondo reale si può vedere nella Figura 1.1.

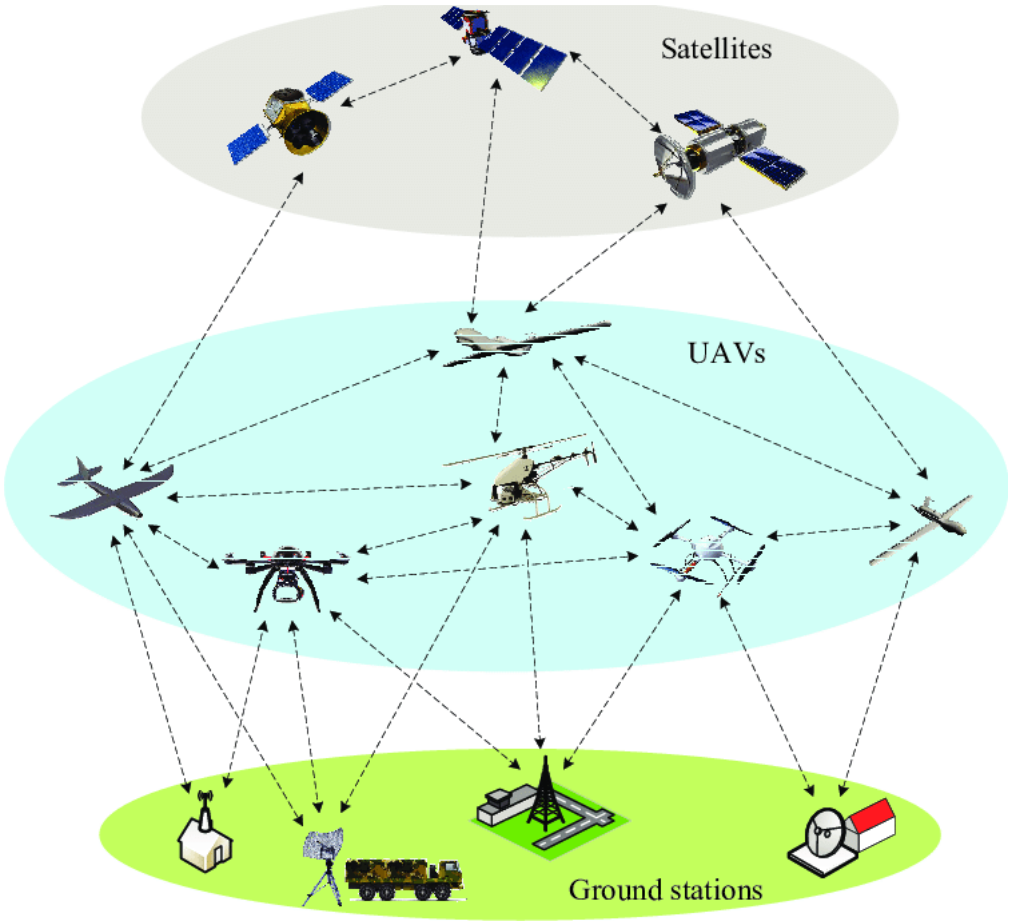

Figura 1.1: Scenario: UAV Swarm Network, un esempio di una rete di UAV eterogenea

1.3.1 Architettura Centralizzata

L'architettura di una rete di UAVs può essere centralizzata, cioè ogni nodo è collegato direttamente alla Ground Station e tutte le comunicazioni passano da li, sia quelle UAV to Infrastructure sia UAV to UAV. Il problema principale di questo tipo di architettura è che la GS diventa il collo di bottiglia dell'intera rete perché, nel momento in cui si guasta nessuna comunicazione sarà più possibile. Inoltre, questo tipo di architettura impone ad ogni UAV di non superare la distanza massima di copertura del segnale dalla GS, limitando quindi la zona operativa della missione. La figura 1.2(a) mostra questo tipo

di architettura.

Il collo di bottiglia della rete e l'impossibilità di ampliare il territorio coperto dagli UAVs oltre il limite imposto dal raggio di copertura, rendono di fatto questa architettura inutilizzabile in contesti ampi e non adatta ad una vasta rete di droni.

1.3.2 Architettura Decentralizzata

Questa topologia differisce da quella precedentemente descritta perché non tutti gli UAVs sono connessi alla GS, infatti alcuni di essi sono connessi solamente ad un loro vicino e dovranno passare per quel nodo per poter comunicare con la stazione di terra. Il vantaggio principale di questo tipo di architettura è che la GS non presenta più un collo di bottiglia in tutte le comunicazioni. Dal punto di vista della copertura, un UAV può non essere nel range della GS, ma ciò non vuol dire che è tagliato fuori da ogni comunicazione, perché può connettersi al nodo più vicino a lui della rete e passare per quest'ultimo quando vuole comunicare con il centro di controllo. In questo modo si possono ottenere numerosi vantaggi quali la scalabilita della rete, la possibilità di coprire un territorio più vasto ed inoltre, alcune comunicazioni UAV to UAV avvengono senza generare traffico che passa dalla GS.

Sono state individuate tre tipi di architetture decentralizzate ampiamente analizzate in (1):

- **UAV ad-hoc network**
- **Multi-group UAV ad-hoc network**
- **Multi-layer UAV ad-hoc network**

Ognuna di queste ha una particolare topologia e si differenzia dalle altre per numero di UAVs coinvolti e per le missioni a cui è applicabile.

UAV ad-hoc network

La figura 1.2(b) mostra l'architettura decentralizzata ad-hoc. Come si evince dall'immagine un solo UAV è collegato direttamente alla GS, mentre tutti gli altri soltanto al proprio vicino. L'unico nodo collegato alla GS fa da gateway e tutte le comunicazioni tra gli altri nodi e la stazione di terra passano da lui, mentre quelle infra-UAVs sono gestite autonomamente all'interno del gruppo. Questo tipo di rete è utile per missioni in cui la zona interessata è piccola, perché il numero di nodi all'interno del gruppo non può essere troppo elevato. Il punto critico di questa architettura è sicuramente l'UAV di gateway, perché se andasse fuori uso, tutte le comunicazioni con la GS andrebbero perse.

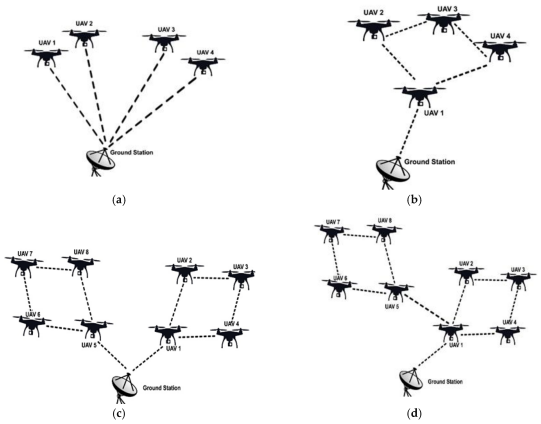

Figura 1.2: Esempi di UAVs Networks.

Multi-group UAV ad-hoc network

L'architettura multi-group presenta più UAVs di gateway, o capo gruppo, ognuno collegato alla GS (figura 1.2(c)). Ognuno di essi è collegato ad un gruppo di altri nodi, che per comunicare con la stazione di terra passano necessariamente da lui. Questo tipo di architettura presenta quindi una divisione del numero totale dei nodi in gruppi, dove ad ognuno è assegnato un capo, il quale sarà collegato direttamente alla GS. Le comunicazioni infra

gruppo sono gestite direttamente dagli UAVs interessati, quelle con la GS invece passano dal gateway, mentre per le comunicazioni tra UAVs di gruppi differenti, i messaggi devono passare necessariamente dalla stazione di terra. Il problema principale di questa architettura è che tutti i capigruppo sono collegati direttamente alla GS, ed il primo livello si presenta quindi come una rete centralizzata, con punto critico la GS. Per questo motivo non è consigliabile utilizzare questo tipo di architettura per gestire applicazioni che richiedono un elevato numero di nodi.

Multi-layer UAV ad-hoc network

La figura 1.2(d) mostra l'architettura multi-layer che presenta una divisione in gruppi del numero totale dei nodi, con soltanto un UAV collegato direttamente alla GS. Ogni capo gruppo è collegato a quello del gruppo precedente, andando a creare una divisione in livelli, dove ogni gruppo diventa uno strato. La GS non è più al centro di ogni comunicazione extra gruppo, perché il messaggio di un nodo che vuole comunicare con un componente di un altro gruppo passa da un UAV di gateway all'altro fino ad arrivare al gruppo dove si trova il nodo destinatario. Il carico di lavoro per la GS è dunque minimo perché riceve soltanto i messaggi a lei indirizzati, direttamente dal UAV di gateway del primo strato.

Questo modello è il migliore per gestire un ampio numero di nodi ed è adatto per il collegamento di molti UAVs in una FANET (Flying Ad-Hoc Network). Con l'aiuto del suo schema multi-hop, questa architettura di comunicazione può fornire copertura estesa per la trasmissione dei dati. è anche possibile aggiungere nuovi gruppi a catena, andando ad espandere l'area operativa della missione. Inoltre è preferita rispetto alle altre a causa della sua natura decentralizzata ed è più flessibile nel fornire una rete di comunicazione "on-the-fly", principalmente per la sua robustezza nei confronti di un singolo punto di errore.

1.4 FANET

Quando un UAV esce dalla zona di copertura della GS generalmente si disconnette dall'intero sistema. In una FANET questo non deve succedere, infatti i vari nodi sono collegati tra di loro in una rete ad-hoc che permette ad un UAV che non rispetta la distanza massima dalla GS di essere ugualmente collegato all'intero sistema. FANET può essere visto come una forma speciale di MANET e VANET, con alcune differenze illustrate nella Fig. 1.3. Tuttavia, ci sono anche alcune differenze tra FANET e le altre reti ad hoc:

1. **Il grado di mobilità** dei nodi FANET è molto più alto rispetto a quello di MANET o VANET, perché in questo caso, i device volano nel cielo, coprendo uno spazio che si estende molto di più in altezza.

2. Data la mobilità dei nodi, **la topologia della rete cambia più frequentemente.**

3. **Le distanze tipiche** tra i nodi FANET sono più **larghe** rispetto a quelle di MANET e VANET.

Tutto ciò colpisce di conseguenza i collegamenti radio, i circuiti hardware e fisici e determina l'esigenza di equipaggiare i vari nodi della rete con diverse antenne e/o sensori a seconda dello scenario applicativo. In genere però, servono due tipi di antenne installate su ogni UAV, quella per comunicare con i nodi adiacenti, che si trovano ad una breve distanza e quella per la comunicazione con la GS, che invece può trovarsi anche a molti km dalla zona applicativa della missione.

Come descritto in (2), non esiste ancora uno standard che definisce il tipo di tecnologia da utilizzare per entrambe le comunicazioni, infatti la scelta viene lasciata a chi progetta il sistema. Esistono diversi modelli che differiscono per la tecnologia di comunicazione utilizzata e per lo scenario applicativo consigliato, ognuno con i suoi pro e i suoi contro.

Nel capitolo seguente verrà descritto lo stato dell'arte per quanto riguarda la comunicazione U2I e U2U in FANET, soffermandosi particolarmente sulle applicazioni LTE e sulla comunicazione ad-hoc con questo tipo di tecnologia.

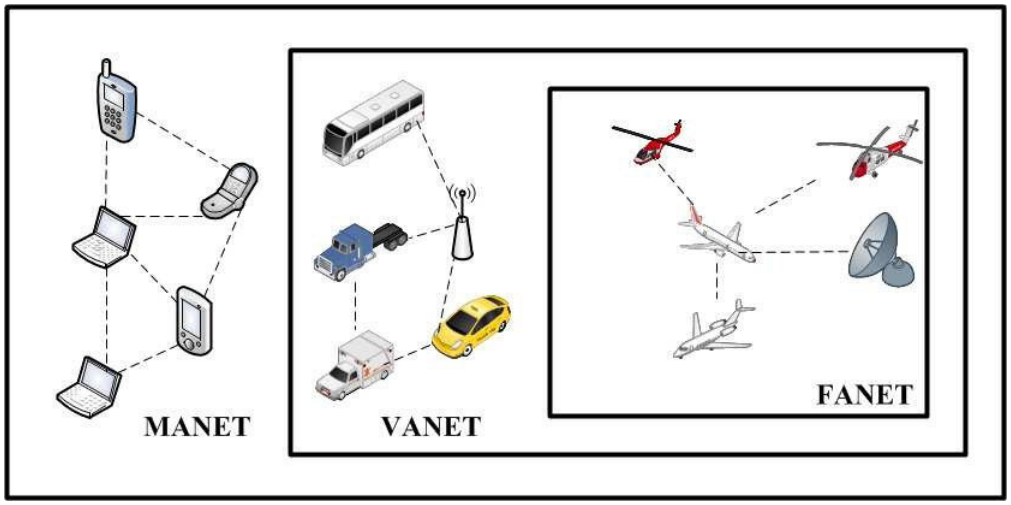

Figura 1.3: MANET, VANET e FANET.

Capitolo 2

Modelli di comunicazione per la rete FANET

In questo capitolo verranno introdotte le principali tecnologie di comunicazione per la rete FANET, sia per quanto riguarda quella U2I (UAV-to-Infrastructure), sia per U2U (UAV-to-UAV), concentrandosi particolarmente sulla tecnologia LTE Side-Link, con modelli che ne costituiscono lo stato dell'arte.

2.1 Cosa si intende per "Modello"

Una rete ad-Hoc FANET è caratterizzata da molteplici UAVs che comunicano fra loro ed eventualmente un solo piccolo sottogruppo di essi ha la capacità di comunicare con la stazione di terra, se presente. Questo tipo di architettura necessita di due tipologie di comunicazioni che sono U2U e U2I. Esistono anche FANET che operano solo con comunicazioni infra-UAV, in questo caso la tecnologia di comunicazione con la stazione di terra non é necessaria.

Ogni nodo della rete si trova quindi a una distanza considerabile breve da ogni altro componente, mentre un UAV di gateway può trovarsi anche a molti KM dalla GS. Grazie a questa topologia di rete, i nodi del gruppo possono essere equipaggiati soltanto con un'antenna a corto raggio, che in certi casi è molto più leggera rispetto alla controparte a lungo raggio ed inoltre consuma meno ed è più economica, mentre un UAV di gateway deve disporre di tutte e due le antenne. In questo modo vengono ottimizzati i costi e l'efficienza energetica dell'intero sistema.

Per modello si intende quindi la scelta di una determinata tecnologia per ognuno dei due tipi di comunicazioni, cercando di massimizzare l'efficienza dell'intero sistema. Non esiste uno standard ben definito per FANET, ma ogni soluzione che viene presa è studiata per lo scenario specifico.

2.2 Tecnologie di comunicazione

Le comunicazioni a lungo raggio richiedono tecnologie che permettono di trasportare il segnale radio da decine di metri fino a diversi km di distanza, per collegare i vari UAVs di gateway alla stazione di terra. In una FANET tutti i membri del gruppo generano dati che dovranno essere poi trasferiti alla GS. A seconda del volume di traffico generato e della dimensione di quest'ultimo, una tecnologia può essere più adatta rispetto ad un'altra per la sua larghezza di banda.

Inoltre, altri fattori possono influenzare la scelta come la dimensione e il peso dell'antenna che andrà montata sull'UAV, il consumo energetico e le frequenze utilizzate. A questo proposito, la scelta deve anche essere fatta prendendo in considerazione lo spettro a cui la tecnologia lavora, perché alcune di esse necessitano di una banda licenziata.

Per quanto riguarda le comunicazioni a corto raggio, cioè quelle infra-gruppo (U2U), la distanza da coprire è ridotta rispetto alla controparte U2I. Siccome ogni UAV del gruppo necessita di un'antenna a corto raggio, è importante che questa sia leggera e consumi poco, in modo da ottimizzare l'efficienza energetica del nodo.

Allo stato dell'arte le tecnologie di comunicazione si differenziano in due grandi categorie, descritte in (3):

- **Non-3GPP Technologies**
- **3GPP Technologies**

2.2.1 Non-3GPP

Per Non-3GPP si intendono tutte quelle tecnologie che non hanno uno spettro di frequenza licenziato. Queste tecnologie solitamente presentano una latenza elevata e sono meno sicure rispetto alla controparte, ma sono meno costose e più semplici da implementare. Inoltre, sono caratterizzate da copertura del segnale ridotta e velocità bassa.

Alcuni esempi possono essere:

- **Wi-Fi**
- **Lora**
- **Zig-Bee**
- **Bluetooth**

Queste tecnologie a frequenza non licenziata possono essere utilizzate per creare dei modelli che lavorano su una distanza medio-corta, in letteratura

infatti ci sono diversi studi che le utilizzano sopratutto per comunicazioni U2U, ma anche per U2I (Wi-Fi per esempio).

2.2.2 3GPP

Le tecnologie 3GPP invece utilizzano bande licenziate, sono più sicure e offrono una copertura molto più vasta. Esse sono di tipo cellulare, alcuni esempi sono:

- **LTE**
- **4G/5G/B5G System**

Queste tipologie possono essere utilizzate sia per U2U che per U2I, inoltre garantiscono un elevato throughput e sono adatte per applicazioni come la video sorveglianza, che necessitano di trasportare una mole di dati non indifferente.

La principale caratteristica del 5G è quella di portare un notevole miglioramento per quanto riguarda la latenza, infatti questa tecnologia fa parte della URLLC (Ultra reliable low latency communication), il che va a migliorare notevolmente le performance e permette di costruire swarm di UAV time-critical.

2.3 LTE

La tecnologia LTE è la quarta generazione per i sistemi ad accesso mobile a banda larga ed è stata standardizzata 3GPP nel 2008. Si tratta di un mezzo di comunicazione dalle performance elevate e da una latenza medio bassa, utile per trasmettere una quantità ingente di dati visto il throughput alto. La copertura del segnale è molto vasta, infatti un nodo può essere distante dalla cella idealmente fino a 100km, le performance ottimali si raggiungono però fino ad un massimo di 5.

Questa tecnologia è utilizzata in tutto il mondo per le comunicazioni cellulari, anche se lo standard della generazione successiva sta sempre più

prendendo piede. In ambito droni molteplici studi sono stati compiuti per l'utilizzo di LTE sia per la comunicazione U2I, visto la portata di questo tipo di tecnologia, sia per la comunicazione U2U. In particolare esistono due tipi di comunicazioni Device-2-Device LTE, il primo e più semplice prevede l'utilizzo della cella per la sincronizzazione dei due dispositivi, cioè per la scelta del canale su cui trasmettere. In questo modo entrambi i dispositivi devo essere nel raggio di copertura. Il secondo metodo invece, non si serve della cella per la scelta del canale, bensì utilizza un metodo di comunicazione diretto tra i due nodi. Ciò permette ai due dispositivi di non dover per forza essere nel campo di copertura di una cella e di poter comunicare ovunque essi siano. Questo tipo di tecnologia prende il nome di LTE Side-Link (4).

2.3.1 Side-Link

"Side-link" è una tecnologia cellulare che consente la trasmissione diretta tra dispositivi con o senza coinvolgimento della stazione base. Ci sono due elementi fondamentali alla base del Side-Link che sono:

- **Discovery**
- **Communication**

Per Discovery si intende la capacita del dispositivo di scansionare l'area circostante e rilevare altri dispositivi nelle vicinanze attraverso la propria interfaccia radio Side-Link (PC5).

La nozione di prossimità non dovrebbe essere pensata solo rispetto alle misure di distanza, ma può anche essere basata sulla qualità del canale radio che la coppia di dispositivi utilizza per la comunicazione oppure sulla qualità del segnale, sul ritardo o su altri parametri. I criteri per definire la prossimità sono quindi piuttosto soggettivi e non si può definire una nozione letterale di prossimità.

Per quanto riguarda invece la comunicazione, nelle modalità più evolute i due dispositivi si accordano sul canale e non necessitano della presenza di

una cella. Ciò permette ad entrambi di essere indipendenti e poter funzionare sia sotto copertura della rete LTE sia dove non c'è campo. Un esempio dell'architettura di rete con Side-Link si può trovare nella Figura 2.1. Ci sono

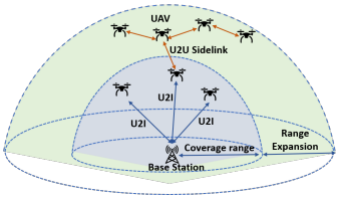

Figura 2.1: UAVs Network Side-Link

diversi studi che analizzano l'efficienza e le performance del Side-Link applicato a reti Ad-Hoc di dispositivi mobili, quali veicoli oppure droni. Come ad esempio lo studio dei miglioramenti apportati alle comunicazioni broadcast (5), oppure lo studio di diverse strategie di scheduling dei pacchetti in base alla posizione (6).

Capitolo 3

UAV Network Side-Link

In questo capitolo verranno analizzati degli studi che compongono lo stato dell'arte per quanto riguarda la comunicazione Side-Link. Inizialmente verranno discusse le varie modalità esistenti e poi saranno presi in esame alcuni studi che propongono soluzioni alternative per alcuni aspetti critici di questa tecnologia e ne valutano le performance.

3.1 U2U,V2X e D2D

L'introduzione della tecnologia Side-Link ha dapprima rivoluzionato le comunicazioni terrestri tra veicoli e poi quelle aeree, portando nuove soluzioni e vantaggi. Un modulo LTE montato su un drone deve essere in grado di gestire sia la comunicazione con la stazione di terra che quella con i nodi vicini. A differenza delle comunicazioni tra veicoli terresti, ci sono diverse caratteristiche che distinguono il nuovo modello dai precedenti, andando ad aumentare le difficoltà di progettazione e gli obbiettivi da raggiungere nella definizione di una rete di questo tipo.

Le principali differenze si trovano in:

1. **Dimensione dell'ambiente**
2. **Mezzo di propagazione del segnale**
3. **Ottimizzazione nella pianificazione del percorso**
4. **Vincoli di alimentazione**

Gli UAV operano in un area 3D differente rispetto a quella dei dispositivi terresti, dove i ricevitori non superano qualche metro d'altezza. I droni infatti operano a diverse altitudini che possono variare a seconda della missione assegnata, per questo motivo è necessario prestare attenzione nel momento in cui si caratterizzano le performance del Side-Link applicato alle comunicazioni U2U.

Siccome i dispositivi aerei si trovano ad un altitudine maggiore rispetto a quelli terresti, l'ambiente radio cambio drasticamente. La probabilità che un'altra antenna causi un interferenza nella comunicazione con la stazione di base è molto più alta perché gli UAVs volano molto al disopra di essa. Gli effetti di interferenza che affliggono le comunicazioni aeree sono quindi da ritenere molto diversi rispetto a quelli terrestri. Inoltre, esiste anche un effetto di correlazione spazio-temporale sul collegamento U2U dato dalla diversa velocità di ogni UAV durante la missione. Le caratteristiche di questo

canale di collegamento sono quindi più dinamiche e variano nel tempo rispetto a quelle degli UE (User Equipments) di terra.

Gli UAVs, a differenza dei veicoli terrestri, pianificano il loro percorso e ottimizzano la loro traiettoria in base alla missione che devono svolgere e al territorio che devono monitorare. Un'altra differenza è che gli UE terrestri non considerano il loro movimento come un fattore da ottimizzare, visto che normalmente solo pilotati da un conducente umano. Il Side-Link deve quindi considerare anche questi aspetti, oltre alla missione.

Ogni UAV è alimentato da una propria batteria che garantisce il funzionamento dei motori oltre a quello dei dispositivi di comunicazione installati. Dato il peso contenuto di ogni unità aerea e le piccole dimensioni, anche le batterie non possono essere grandi e capienti, quindi l'autonomia in volo non è solitamente molto elevata. Perciò la carica della batteria rappresenta un collo di bottiglia nelle comunicazioni aree.

3.2 Physical Layer

LTE Side-Link (7) utilizza "single-carrier frequency-division multiple access" e supporta i canali a 10 e 20 MHz. Tutti i canali sono suddivisi in sub-frames, resource blocks (RB) e sub-channels. I sub-frames sono lunghi 1 ms, come l'intervallo di tempo di trasmissione TTI. Un RB è la più piccola unità di risorse di frequenza che può essere assegnata a un utente. LTE ha un'ampiezza di frequenza di 180 kHz (12 sotto-portanti da 15 kHz) e definisce i sub-channels come gruppo di RB nello stesso sub-frame, dove il numero di RB può però variare. I dati vengono trasmessi in transport-block (TB) sui "Physical Side-Link Shared Channels" (PSSCH), e i "Side-Link Control Information messages" (SCI) sono trasmessi sui "Physical Side-Link Control Channels" (PSCCH).

Un TB contiene un pacchetto completo da trasmettere, ad ognuno di questi è legato un SCI che deve essere trasmesso anch'esso ed è stato assegnato al momento dello scheduling del pacchetto sulla rete. Lo SCI include MCS

(modulation and coding scheme, usato per la trasmissione del TB), i RBs utilizzati e lo RRI (resource reservation interval) che viene utilizzato per il "semipersistent scheduling (SPS)". La corretta ricezione del pacchetto SCI da parte del nodo destinatario è essenziale per ricevere e decodificare il TB a lui indirizzato, perciò il TB e lo SCI associato devono essere trasmessi nello stesso sub-frame.

3.3 Tipi di comunicazione Side-Link

Esistono 4 diverse modalità (8) con cui la tecnologia Side-Link può funzionare, ognuna dotata di caratteristiche ben precise.

La Mode 1 e la 2 comportano, a differenza della 3 e della 4, la presenza di una stazione di terra che si chiama "eNodeB" (hardware connesso alla rete di telefonia mobile che comunica direttamente in modalità wireless con gli UE, come una stazione base di ritrasmissione (BTS) nelle reti GSM). Nel primo caso i due dispositivi che vogliono parlare utilizzano risorse radio dedicate, scelte direttamente dall'eNodeB, in mode 2 invece selezionano in modo casuale le risorse radio da un pool rilasciato precedentemente dalla stazione di terra. Entrambe le modalità condividono la stessa struttura di allocazione delle risorse, in cui la trasmissione dei dati è programmata entro il cosiddetto "Physical Side-Link Control Channel (PSCCH) period". Entro questo periodo, viene determinato un insieme di subframe per la trasmissione PSCCH (parte azzurra nella Fig.3.1(a)) e un secondo insieme per il "Physical Side-Link Shared Channel" (PSSCH). (parte gialla nella Fig.3.1(a)).

Il PSCCH corrispondente per un determinato PSSCH viene sempre inviato prima dei dati PSSCH. Il PSCCH contiene le informazioni di controllo del Side-Link (SCI), chiamate anche scheduling assignment (SA), che vengono utilizzate dal ricevitore per conoscere l'occupazione delle risorse radio PSSCH. In entrambe le modalità, la SCI è configurata nel formato 0 e viene trasmessa in modo identico in due differenti subframe. Questa ritrasmissione è sempre necessaria a causa della mancanza di un canale di feedback nella

comunicazione Side-Link. Il ricevitore rileva sempre la SCI provando tutte le possibili risorse PSCCH. Il PSSCH TB (transport block) viene trasmesso quattro volte in quattro subframe consecutivi all'interno del pool di subframe, consentendo al dispositivo ricevitore di implementare la richiesta di ripetizione automatica ibrida (HARQ) a loop infinito combinando le quattro versioni di ridondanza del PSSCH TB.

Allo stesso modo, la differenza tra le Mode 3 e le Mode 4 è che nella prima l'allocazione delle risorse per la trasmissione dati è assistita dalla eNodeB mentre nella seconda i dispositivi scelgono le risorse radio in modo autonomo. Tuttavia, le Mode 3 e 4 condividono una struttura completamente diversa rispetto a quella delle Mode 1-2 descritta sopra per allocare PSCCH e PSSCH. In primo luogo, non esiste un periodo PSCCH per distribuire la trasmissione di entrambi i canali fisici in periodi temporali diversi. Al contrario, i canali PSCCH e PSSCH sono separati nel dominio della frequenza. La griglia delle risorse è suddivisa in sub-bands o sub-channels in cui i primi RB (resource blocks) (frequenze più basse) di questi sub-channels formano il pool PSCCH (parte azzurra nella Fig.3.1(b)) e gli altri RB il PSSCH pool (parte gialla nella Fig.3.1(b)). Due SCIs identici (in formato 1) e i corrispondenti TBs PSSCH vengono inviati nello stesso sub-frame per ovviare alla distorsione del canale. Nelle Mode 3-4, un TB può essere inviato una o due volte. In caso di due tentativi di trasmissione, viene utilizzato un altro sub-frame con la stessa struttura: due SCIs e il corrispondente TB PSSCH. In questo caso, il ricevitore implementa anche HARQ.

La Fig.3.1 illustra diversi scenari all'interno delle diverse modalità di comunicazione e permette di evidenziare le principali differenze tra le Mode 1-2 e le Mode 3-4. Nell'esempio illustrato in figura, tre pacchetti con dimensioni diverse arrivano da layer superiori in istanti differenti. Le frecce all'interno della griglia mostrano le occupazioni PSSCH determinate da un SCI. Uno dei principali svantaggi delle Mode 3-4 rispetto alle Mode 1-2 è l'efficienza spettrale poiché i sub-channels non sono sempre pieni di dati durante l'utilizzo. Si osserva in Fig.3.1(b) che, nella trasmissione del secondo

TB (pacchetto da 7 byte), i dati (parte arancione) occupano solo una breve porzione dell'intero sub-channel. Gli altri RB del sub-channel non sono usati in questa trasmissione e non possono essere usati da altri dispositivi con la stessa configurazione di griglia senza collidere, anche se c'era spazio sufficiente per entrambi per trasmettere nello stesso sub-channel. Ciò è dovuto alla granularità del sub-channel nell'allocazione delle risorse nelle Mode 3-4, in cui la trasmissione dei dati inizia sempre alle frequenze più basse di esso.

Ciò limita la flessibilità di scheduling poiché differenti dispositivi possono essere allocati in un minor numero di posizioni differenti rispetto alle Mode 1-2, dove la granularità di schedulazione è il RB (Resource Block). Pertanto, non vi è solo un'efficienza spettrale inferiore nelle Mode 3-4 rispetto alle Mode 1-2, ma vi è anche una maggiore probabilità di collisione quando più dispositivi vicini stanno trasmettendo contemporaneamente e i sub-channels sono parzialmente vuoti. Il vantaggio principale delle Mode 3-4 è la latenza poiché un pacchetto arrivato dai livelli superiori viene immediatamente schedulato in uno qualsiasi dei sub-frame successivi disponibili all'interno del pool, come illustrato nella Fig.3.1(b). Invece, nelle Mode 1-2 la trasmissione del pacchetto potrebbe dover attendere l'inizio del successivo periodo PSCCH, come illustrato per il secondo pacchetto in Fig.3.1(a). Inoltre, le Mode 3-4 introducono simboli DMRS (Demodulation Reference Signal) aggiuntivi per gestire gli effetti Doppler elevati in scenari ad alta mobilità. Per queste ragioni, le Mode 3 e 4 sono state proposte nella Release 14 del 3GPP come modalità di comunicazione per le applicazioni V2X.

- **Adjacent Mode**: Il TB e lo SCI corrispondente sono trasmessi in RBs adiacenti. Lo SCI occupa i primi due RBs del primo sub-channel e il TB invece viene trasmesso sui RBs appena successivi e può occupare diversi sub-channels, ciò dipende dalla sua dimensione. Se è necessario occupare altri sub-channels allora il TB occuperà anche i primi due RBs di tutti quelli interessati alla trasmissione.

- **Non-Adjacent Mode**: I RBs sono suddivisi in pool, di cui uno dedicato per trasmettere gli SCI (che occupano sempre due RBs), mentre

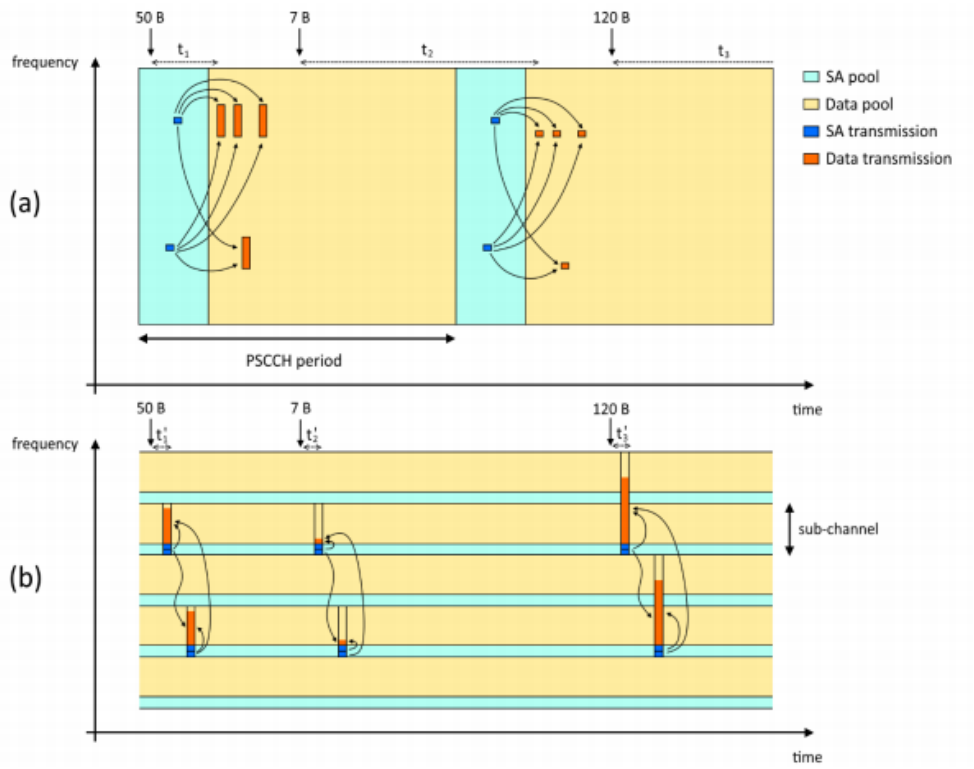

Figura 3.1: Allocazione delle risorse nelle quattro modalità del Side-Link: a) Mode 1-2 b) Mode 3-4.

il secondo viene riservato alla trasmissione dei TBs ed è suddiviso in sub-channels. i TBs possono essere trasmessi utilizzando QPSK oppure 16-QAM, mentre gli SCIs vengono sempre trasmessi utilizzando QPSK.

Le due modalità sono ampiamente descritte in (9).

3.4 Sensing-Based SPS

La scheduling delle risorse (10) viene eseguito su uno spazio bidimensionale, dove i due assi sono il tempo e la frequenza. La griglia delle risorse è suddivisa in sub-channels in frequenza e sub-frames nel tempo. Sull'asse della frequenza, la granularità dell'allocazione delle risorse è un sub-channel.

Ogni blocco verticale nella griglia è definito da uno o più sub-channels e un singolo sub-frame (Fig. 3.2). In SPS, ad un gruppo di pacchetti equidistanti sull'asse del tempo vengono assegnati gli stessi sub-channels. Lo spazio tra i pacchetti è chiamato "Resource Reservation Interval" (RRI), i cui valori tipici sono 20 ms, 50 ms oppure 100 ms.

Per determinare quale risorsa un dispositivo può utilizzare per il gruppo di pacchetti successivo, ci si basa sul rilevamento dell'uso delle risorse da parte dei dispositivi vicini nel passato più recente. Ogni veicolo monitora 1000 sub-frames nell'immediato passato, questa finestra di tempo è chiamata "Sensing Window" e serve per identificare le risorse che sono state utilizzate da altri dispositivi. L'utilizzo o meno è determinato dal campo RRI dello SCI del pacchetto ricevuto (Fig. 3.3) che indica se il pacchetto successivo utilizzerà o meno lo stesso sub-channel nell'intervallo dato. Lo standard 3GPP utilizza un codepoint a 4 bit nello SCI. Ad esempio, "1010" indica che un altro pacchetto seguirà in 100 ms sullo stesso sub-channel, mentre "0000" indica che non avverranno altre trasmissioni in quel gruppo.

Se la risorsa riservata al pacchetto immediatamente precedente da un altro veicolo nel campo RRI ha un RSRP (Reference Signal Received Power) maggiore di una soglia preimpostata e rientra nella Selection Window del nostro dispositivo (Zona gialla Fig. 3.2), viene esclusa dal pool di risorse disponibili. In caso contrario viene controllato l'RSSI medio delle risorse precedenti nella Sensing Window e se è inferiore ad una certa soglia, viene inclusa. Infine, tutti i sub-frames utilizzati dal dispositivo nella Sensing Window vengono proiettati nella Selection Window per essere esclusi come non selezionabili. Questo perché i sub-frames in questione non potevano essere rilevati dal nostro dispositivo a causa della trasmissione di tipo half-duplex e potrebbero esserci altri veicoli che stavano trasmettendo e che potrebbero tutt'ora continuare a farlo durante la Selection Window.

Dopo aver escluso le risorse non idonee, quelle rimanenti vengono chiamate gruppo candidato SA. Se la dimensione di questo gruppo è minore del 20% delle risorse analizzate nella Selection Window, viene alzata la soglia

RSRP di 3dB e viene ripetuto il processo di filtraggio. Nel passaggio successivo vengono ordinate le risorse in base al RSSI e ne vengono scelte solo il 20% per creare un nuovo gruppo, chiamato SB. Il layer superiore dell'architettura selezione una risorsa casualmente da questo gruppo. Anche il numero di volte in cui il sub-channel selezionato viene utilizzato senza ulteriore segnalazione viene scelto casualmente nell'intervallo [C1, C2] e viene chiamato "Reselection Counter" (RC). [C1, C2] vengono scelti in base alla Tab.3.1.

RRI	**[C1,C2]**
100ms	[5,15]
50ms	[10,30]
20ms	[35,75]

Tabella 3.1: Selezione di [C1,C2] in base al RRI

Ogni pacchetto trasmesso di un gruppo decrementa RC di uno. Quando RC raggiunge 0, dovrebbe essere pianificato l'invio del prossimo gruppo di pacchetti, che viene chiamato "Resource Reselection", con la probabilità di conservazione delle risorse Pk ($0 \leq Pk \leq 0,8$) configurata dai layers superiori. Tuttavia, l'algoritmo può decidere di continuare a utilizzare la stessa risorsa per l'invio successivo (vedere Fig. 3.2). Con 1 - Pk, viene scelta una risorsa diversa da SB in base alla procedura di filtraggio discussa sopra.

3.5 Foreseeing Semi-Persistent Scheduling

Nel processo di Scheduling classico il set delle risorse utilizzabili viene calcolato nella Sensing Window e vengono esclusi circa l'80% dei candidati dall'insieme iniziale. La soglia RSSI viene aumentata di 3dB e un nuovo insieme viene calcolato iterativamente finché non otteniamo un set di risorse con più del 20% di elementi del gruppo iniziale. Pertanto, l'RSSI da solo non può indicare se la risorsa selezionata casualmente dal gruppo è occupata o meno e quindi può succedere che due nodi entrino in collisione.

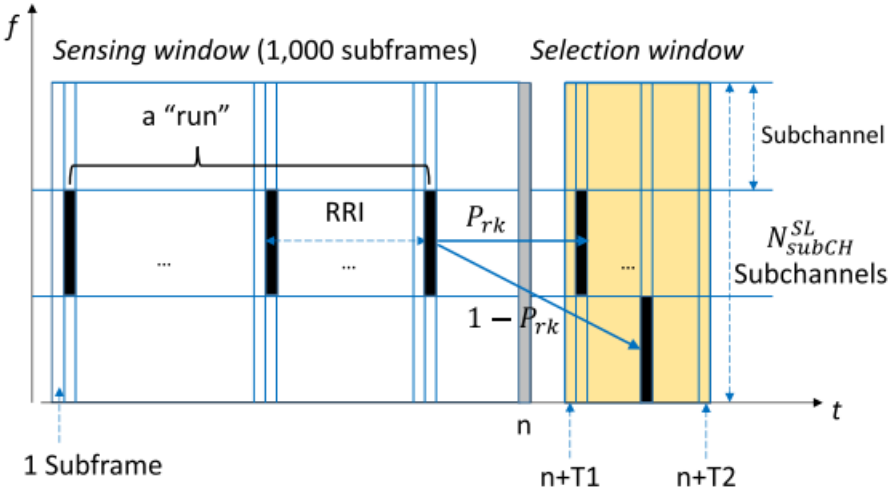

Figura 3.2: Struttura del SB-SPS.

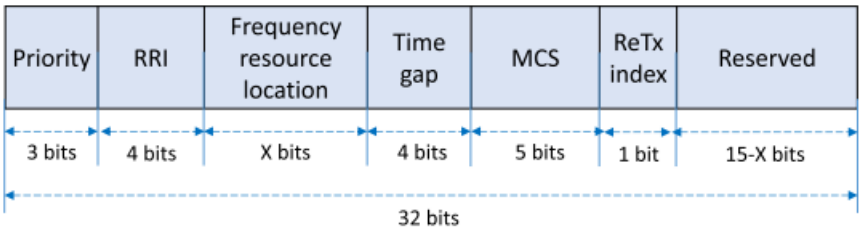

Figura 3.3: 3GPP SCI Format.

Lo SCI trasmesso nel PSCCH nei primi due RB del sub-channel contiene anche informazioni sulle risorse che verranno utilizzate per la ritrasmissione. Partendo dal presupposto che tutti i messaggi inviati siano segnali broadcast, di conseguenza qualsiasi nodo della rete può leggere il valore SCI del messaggio con cui collide. Quindi, l'F-SPS proposto (11) suggerisce che, nella finestra di rilevamento, dopo aver rilevato la collisione, il nodo in questione non misurerà semplicemente l'RSSI, ma decodificherà lo SCI di quello con cui collide e apprenderà le risorse utilizzate per le ritrasmissioni. Inoltre, quando un nodo A subisce una collisione, lo SCI dell'altro nodo non viene solo decodificato e considerato per l'esclusione nella sua futura selezione, ma

anche codificato nel proprio SCI. Successivamente, quando un terzo nodo entra in collisione con il nodo A, questo può decodificare lo SCI di A e avere più informazioni sulle risorse da escludere.

Il grafico in Fig. 3.4 mostra la riduzione del numero delle collisioni utilizzando l'algoritmo F-SPS rispetto al classico SPS.

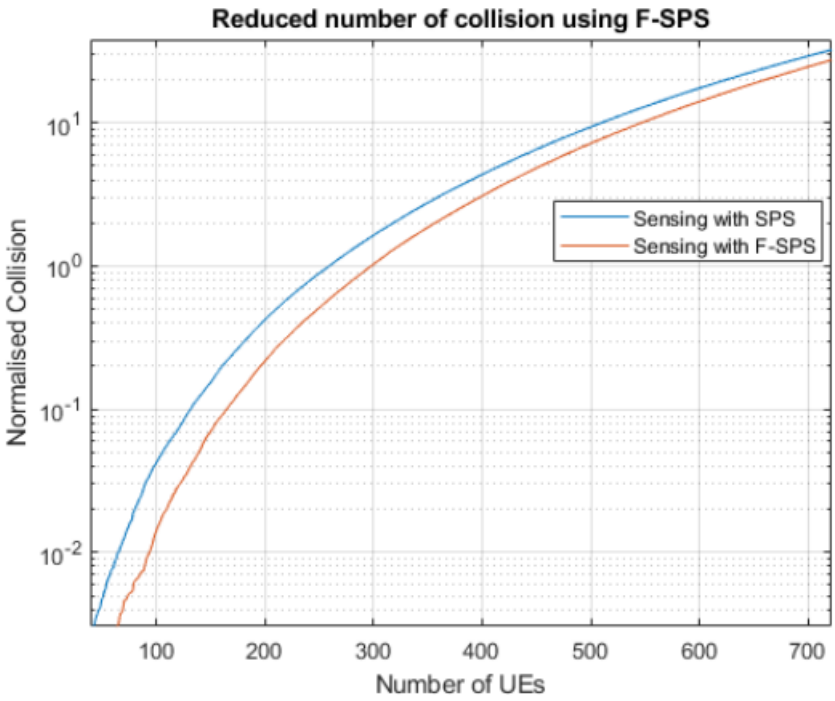

Figura 3.4: Comparazione delle collisioni tra F-SPS e SPS

3.6 Gapped Reservation-augmented SPS

L'obiettivo principale di questa proposta (10) è potenziare SPS con un meccanismo di prenotazione aggiuntivo che presenta tre caratteristiche:

- **Explicit:** la posizione è specificata esplicitamente da una coordinata di frequenza temporale.
- **Early:** la prenotazione può essere effettuata per una location lontana nel futuro, molto prima che avvenga il suo reale utilizzo.
- **Repeated:** le informazioni di prenotazione vengono ripetute molte volte per incrementare l'affidabilità.

La prima caratteristica impone che ogni veicolo informi esplicitamente gli altri con la posizione della risorsa riselezionata. Utilizzando una coordinata tempo-frequenza della risorsa da utilizzare, possiamo prenotare la trasmissione del pacchetto in un momento arbitrario nel futuro e in un diverso sub-channel, se necessario. Al contrario, l'algoritmo SPS standard specifica solo l'intervallo di tempo tra il pacchetto corrente e il successivo, quindi la prenotazione è limitata al canale corrente e al pacchetto successivo. A causa di questa limitazione, la prenotazione SPS non può specificare alcuna informazione sulla posizione della risorsa successiva al momento della riselezione perché non è possibile selezionare un sub-channel diverso.

La seconda caratteristica che emerge dalla prima è che con questo algoritmo la prenotazione può essere effettuata molti pacchetti prima dell'effettiva trasmissione sulla risorsa selezionata, in modo da consentire ai veicoli vicini di prendere una decisione informata quando è il loro turno di prenotare o selezionare una risorsa.

La terza funzionalità è abilitata dalla seconda. Poiché le informazioni sulla prenotazione vengono visualizzate in ogni pacchetto prima della trasmissione effettiva utilizzando la risorsa riservata, una prenotazione anticipata significa un numero maggiore di annunci. Ciò migliora l'affidabilità delle informazioni di prenotazione trasmesse ai vicini.

Nell'articolo in questione viene proposto questo algoritmo e vengono analizzati i vantaggi rispetto a quello standard focalizzandosi sulla diminuzione delle collisioni e sull'aumento del "Packet Reception Ratio".

3.7 Cooperative SPS

Nell'approccio SPS standard, le collisioni tra diversi nodi si verificano mentre le Selection Windows si sovrappongono. Il Cooperative SPS (8) affronta questo problema in modo più efficiente in termini di affidabilità. La proposta consiste nel trasmettere i valori del contatore in ciascuna trasmissione di pacchetti in modo che i vari nodi siano consapevoli dei contatori correnti dei loro vicini.

Quindi, i dispositivi attiveranno la riselezione del contatore se uno qualsiasi dei contatori ricevuti nell'ultimo RRI coincide con il proprio nel momento corrente. In questo modo, si evita che due UE che arrivano al contatore 0 contemporaneamente eseguano la riselezione delle risorse ad istanti di tempo limitrofi, portando ad una Selection Window priva di collisioni.

Si osserva nella Fig. 3.5 che l'approccio proposto raggiunge chiaramente una probabilità di collisione inferiore (0,14% dopo 1000 secondi nello scenario simulato) rispetto all'approccio standardizzato (0,64%). In effetti, tutte le collisioni provengono dai nuovi nodi che entrano nel sistema. Ciò è dimostrato in Fig. 3.6, dove viene considerato uno scenario statico, cioè senza nodi che escono o che si uniscono.

In questo caso, possiamo vedere che la probabilità media di collisione diminuisce durante tutto il periodo di simulazione. Questo perché dopo alcune collisioni all'inizio della simulazione causate da un'inizializzazione random delle risorse, non ci sono più collisioni.

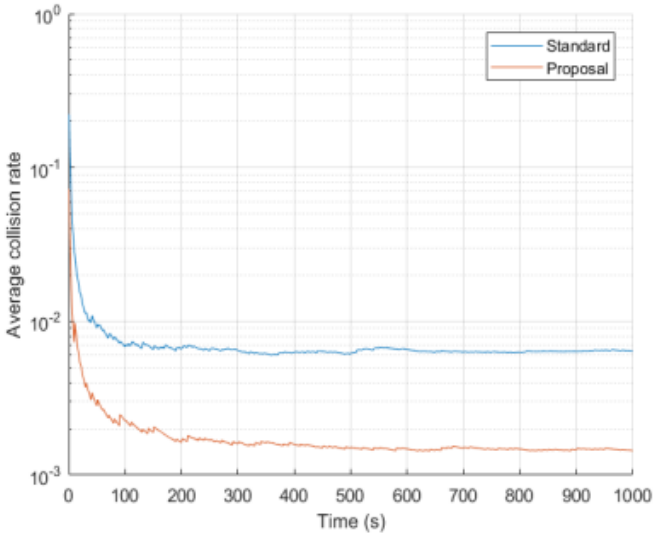

Figura 3.5: Probabilità media delle collisioni in uno scenario dinamico: SPS vs Cooperative SPS

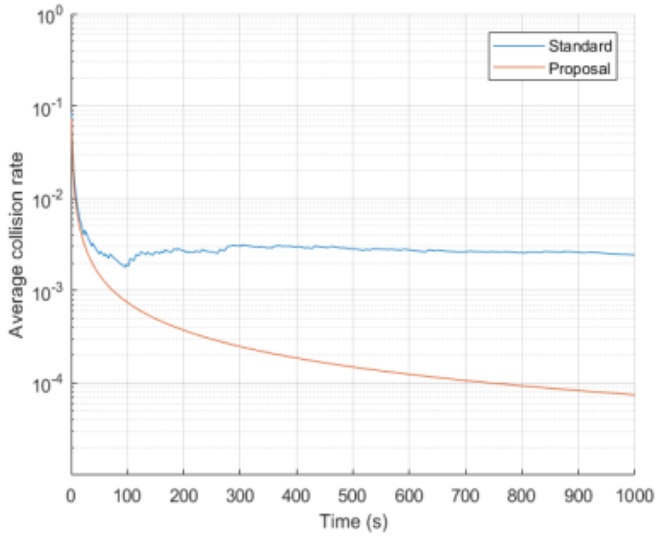

Figura 3.6: Probabilità media delle collisioni in uno scenario statico: SPS vs Cooperative SPS

3.8 Estimation and Reservation Resource Allocation Algorithm

SPS standard non riesce a risolvere i problemi che sorgono quando due o più veicoli scelgono lo stesso sub-channel contemporaneamente, di conseguenza, possono avvenire collisioni tra pacchetti. L'algoritmo in questione gestisce questi errori. ERRA (12) funziona nel seguente modo:

- Il nodo UEi **analizza continuamente** i pacchetti sulla rete e apprende informazioni dalle SCIs decodificate che provengono dagli altri nodi nel suo raggio di copertura.
- Quindi, il nodo in questione salva tutte le posizioni dei pacchetti che hanno lo **stesso valore RC** nel nuovo elenco A.
- La **posizione** della risorsa stimata **viene scelta dall'elenco A** quando RC del nodo raggiunge 5 e quindi viene annunciata la posizione della risorsa successiva, come illustrato nella Fig. 3.7 (UEi, UE3, UE7, UE8).
- L'UEi **controlla costantemente la posizione successiva** predefinita per assicurarsi che sarà disponibile nella riselezione successiva (vedere UE2 nella Fig. 3.7).
- Quando RC del nodo **raggiunge zero**, la posizione della risorsa successiva selezionata viene **utilizzata per la trasmissione** e viene generato il nuovo RC (vedere UE1 nella Fig. 3.7).

È necessaria una piccola estensione per lo SCI (circa 2 byte) per ottenere una maggiore affidabilità nella trasmissione, per l'annuncio della posizione della risorsa successiva quando è necessario, per RC e posizione della collisione se esiste. L'algoritmo in questione non dipende dal processo di "Sensing" e quindi riduce le collisioni dei pacchetti che possono verificarsi a causa della selezione delle risorse.

In questo lavoro viene proposto che RC non sia generato in modo completamente random come S-SPS. Il valore RC di ogni nodo dovrebbe essere compreso tra 6 e 15, ma il nuovo RCi dovrebbe essere anche compatibile con altri valori di pacchetto RC che vengono trasmessi nello stesso sub-frame (UE10, UE11, UE12 e UE13 nella Fig. 3.7). Di conseguenza, in questo modo si evita che i dispositivi trasmettano nello stesso sub-frame preservando la risorsa per la successiva riselezione.

La disponibilità della risorsa successiva o di quella corrente dell'UEi può cambiare, specie quando una nuova UE entra nel raggio di copertura del dispositivo con la stessa risorsa dichiarata come successiva, con conseguente collisione tra pacchetti (vedi UE4 nella Fig. 3.7). I nodi che si accorgono di queste collisioni indicano a tutti gli altri la presenza dell'anomalia andando ad estendere lo SCI. In questo modo tutti i dispositivi sanno quando avviene una collisione durante la trasmissione.

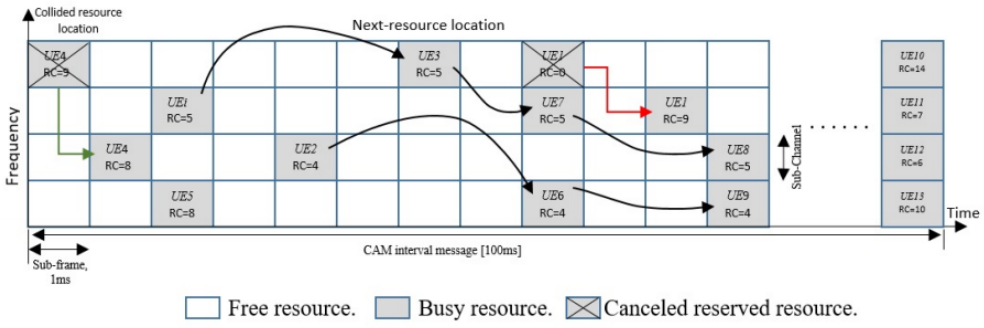

Figura 3.7: Scheduling: Algoritmo ERRA

3.9 Conclusioni

Lo stato dell'arte per quanto riguarda lo Scheduling in FANET propone diverse soluzioni oltre a quelle elencate nelle precedenti sezioni, tutte che puntano a trovare un algoritmo con performance migliori rispetto a quelli precedentemente proposti. Alcuni puntano a migliorare il meccanismo di rilevazione delle collisioni, altri a diminuire la congestione della rete, altri ancora a migliorare il consumo di energia da parte dei vari nodi interessati. Non esiste quindi, per ora, un algoritmo che riesca ad ottimizzare tutti i punti contemporaneamente, perciò l'argomento è ad oggi molto trattato ed ancora in fase di ricerca.

Parte II

Valutazione di Side-Link su UAV Network

Capitolo 4

Modellazione

In questo capitolo viene descritta l'intera modellazione del sistema simulativo proposto, partendo dagli obbiettivi iniziali, fino alla progettazione della rete di UAVs, con tutte le assunzioni fatte, dettate dalla pratica e dagli strumenti in mio possesso.

4.1 Descrizione dell'obbiettivo

Nell'ambito delle comunicazioni tra droni non esiste ancora uno standard ben preciso, visto le diverse potenzialità di ogni tecnologia, legate anche alla topologia della rete di UAVs. L'obbiettivo iniziale di questo progetto è simulare una rete di droni, in cui viene utilizzata la tecnologia Side-Link LTE per la comunicazione U2U. Ogni membro invia dei semplici pacchetti di "Alert" a tutti i gli altri componenti della rete, per un periodo limitato di 60 secondi, per poi analizzare attraverso delle metriche ben precise, la qualità della rete e della comunicazione.

I vari UAV all'interno della rete sono posti a diverse distanze tra loro, ed è quindi possibile che il destinatario di un messaggio non si trovi nel raggio di copertura del mittente e che non riesca a ricevere il messaggio. Per questo motivo è necessario introdurre un meccanismo di routing dei vari pacchetti, in modo tale che il messaggio venga recapitato al destinatario, passando dagli altri UAVs che si trovano nello spazio intermedio tra esso e il mittente. Ogni UAV che il messaggio deve attraversare è un "Hop", a seconda del numero di salti il messaggio subisce un ritardo dovuto appunto all'instradamento dei vari pacchetti.

Ogni componente delle rete deve quindi inoltrare il messaggio ricevuto se non è lui il destinatario. Questa azione viene gestita da un algoritmo di routing proposto e valutato tramite le metriche classiche di qualità della rete, discusse nel capitolo inerente alla validazione.

4.2 Scelta della tecnologia

La prima scelta in fase di modellazione del problema è stata quella della tecnologia di comunicazione, che è ricaduta dapprima su LTE. Sicuramente il mondo delle comunicazioni mobili è strettamente legato alla tecnologia cellulare e allo stato dell'arte, i progetti più innovativi si basano sulla quinta generazione di queste reti. LTE rappresenta invece la quarta generazione, che a differenza della successiva presenta latenza molto più alta e throughput

relativamente più basso. Al momento però non è disponibile un framework avanzato che permetta di utilizzare la modalità Side-Link di questa tecnologia con la quinta generazione della rete mobile, perciò la scelta è ricaduta sulla precedente.

La modalità Side-Link permette la comunicazione D2D diretta e a seconda della versione necessita o meno della copertura della rete cellulare. Infatti, la Mode 4 permette lo scambio di messaggi tra due dispositivi vicini, senza essere nel raggio di copertura di una stazione cellulare ed è una connessione diretta che viene gestita interamente dai due dialoganti. Tutta la parte di gestione del canale e della scelta delle risorse avviene nel UAV mittente.

Gli UAVs della rete dovranno quindi soltanto essere equipaggiati con un'antenna LTE compatibile con Side-Link Mode 4 ed essere in grado di eseguire un'applicazione che gestisca la creazione e il routing dei vari pacchetti.

4.3 Scenario

Lo scenario prevede un gruppo di UAVs che comunica con tecnologia Side-Link. Per creare ed imporre uno schema multi-hop per la comunicazione é stato scelto di disporre i vari componenti del gruppo come una catena, in modo tale che ogni UAV comunichi soltanto con i suoi due vicini, oppure con uno solo nel caso degli estremi. Questa scelta semplifica i passi necessari per gestire il multi-hop, infatti i pacchetti passano da ogni componente della catena, che controlla se sono destinati a lui e in caso contrario, li inoltra.

Ogni UAV deve essere provvisto dei seguenti componenti (Fig. 4.1):

- **Scheda di rete LTE (LteNic)**
- **Applicazione per la gestione dei messaggi**
- **Gestione della mobilità**

Ognuno dei componenti svolge una funzione ben precisa. La scheda di rete si occupa dello scambio dei messaggi, la "Mobility" gestisce il posizionamento

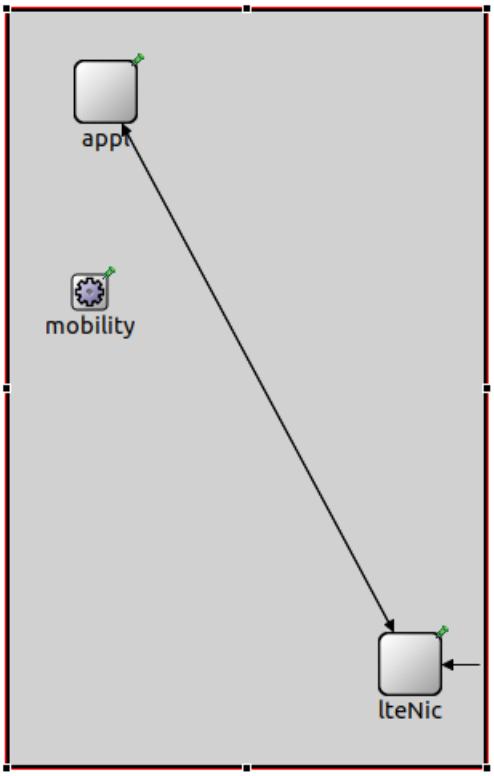

Figura 4.1: Modello di un UAV, application manager, mobility e LteNic

del device in tempo reale e l'applicazione invece si occupa della creazione e della gestione dei messaggi rispettivamente inviati e ricevuti dal device in questione.

La scelta del modello di posizionamento a catena semplifica la gestione dei pacchetti, perché progettare un algoritmo che tenesse conto della posizione in tempo reale dei vari nodi in una rete dinamica e in movimento e che ne gestisse il routing dei pacchetti sarebbe stato veramente complicato e quindi sono state fatte alcune assunzioni. Lo scenario proposto impone la staticità di ogni UAV, ad una distanza ben precisa l'uno dall'altro, utile alla sperimentazione dell'algoritmo di routing proposto.

4.3.1 Posizionamento degli UAVs

Il Signal Power scelto per la scheda di rete é di 10W, che permette al UAV di raggiungere ogni altro dispositivo nell'arco di 1500 metri. Perciò i vari UAVs vengono posizionati ad una distanza di 800m l'uno dall'altro (Fig. 4.2), in sequenza, in modo tale che ognuno di essi possa dialogare soltanto con i suoi vicini, per simulare una catena di "Hop" statica.

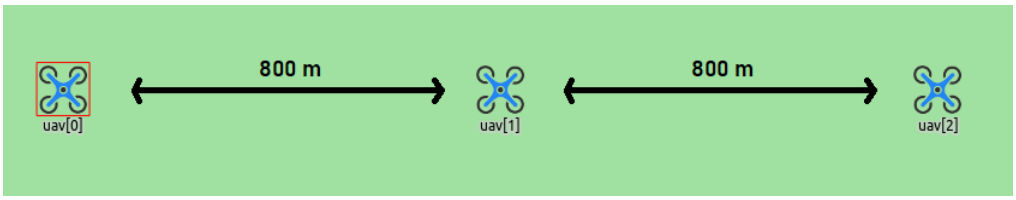

Figura 4.2: Posizionamento degli UAVs nella rete della simulazione

4.3.2 Algoritmo per la gestione dei messaggi

La creazione e la gestione dei messaggi rispettivamente inviati e ricevuti è affidata all'applicazione eseguita su ogni dispositivo. Assumiamo che soltanto il primo UAV della catena possa inviare messaggi di "Alert" e che il destinatario sia sempre l'ultimo. L'applicazione è quindi strutturata come segue:

- **Creazione ed Invio del messaggio di "Alert"**
- **Ricezione o inoltro del messaggio**

La prima fase viene eseguita soltanto nel primo UAV e si occupa di schedulare ad ogni intervallo di tempo specificato dall'utente, l'invio di un messaggio di "Alert" in broadcast con destinatario l'ultimo UAV della catena. Questo messaggio verrà ricevuto da tutti gli UAV nel raggio di copertura del mittente, ma soltanto il vero destinatario elaborerà il contenuto.

La seconda fase invece viene attivata quando un UAV riceve un messaggio dal suo vicino, per prima cosa controlla se è effettivamente lui il destinatario

e in questo caso elabora il messaggio e salva le statistiche. Nel caso contrario invece, viene inoltrato il messaggio sempre in broadcast. Così facendo si crea una catena ed il messaggio viene continuamente inoltrato fino a quando non raggiunge il destinatario.

Quando un device trasmette in broadcast, entrambi i vicini ricevono, quindi è necessario aggiungere un controllo sui pacchetti già inoltrati, in modo tale da non creare cicli infiniti. Per questo motivo è essenziale aggiungere un numero di sequenza, in modo tale da poter tener traccia di tutti i pacchetti presenti nella rete.

Nella Fig. 4.3 è illustrato il semplice procedimento dell'algoritmo.

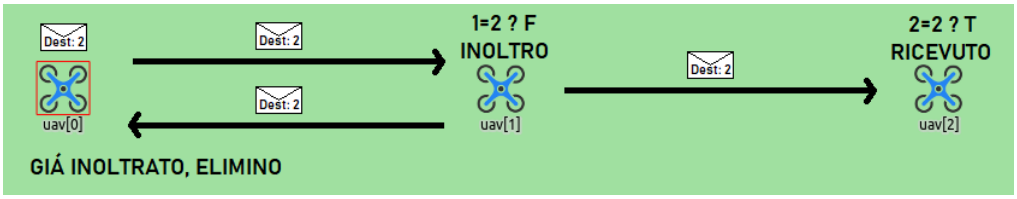

Figura 4.3: Algoritmo di Routing statico con messaggi Broadcast

4.4 Assunzioni pratiche

Per applicare l'algoritmo di routing precedentemente descritto è necessario imporre alcune limitazioni alla rete di UAV in esame. La prima assunzione impone che tutti i dispositivi, dopo aver acquisito la posizione iniziale, la mantengano. Questo è importante perché l'algoritmo proposto ha bisogno di una catena statica di "Hop" e al contrario, servirebbero delle regole molto più complicate per poter permettere il calcolo della posizione del device destinatario in modo dinamico. Ci sono alcuni studi e alcune proposte per algoritmi di questo come il Geographic Mobility Prediction Routing (13), che prevede un protocollo di instradamento di previsione della mobilità geografica efficace per migliorare le prestazioni di instradamento tra UAV. Oppure Greedy Forwarding and Limited Flooding based Routing (14), che divide il processo di instradamento dei pacchetti in FANET nella fase "Greedy" e

nella fase di individuazione del percorso di flooding e utilizza la strategia di ottimizzazione dello Swarm di particelle (PSO) per risolvere il problema della scelta non ottimale dell'inoltro Greedy.

Inoltre, assumiamo che soltanto il primo UAV possa creare dei messaggi, perché altrimenti la logica del numero di sequenza dovrebbe essere ampliata per capire se un messaggio è già stato ricevuto o meno.

Per quanto riguarda invece il modello di UAV proposto, non è stata presa in considerazione la gestione dell'energia, in quanto il test viene condotto interamente sulla qualità della rete e sulle performance della tecnologia di comunicazione in esame, perciò sono state sviluppate soltanto le parti legate alla connettività e alla mobilità.

Capitolo 5

Implementazione

In questo capitolo è fornita una descrizione dettagliata della fase di implementazione del modello simulativo, partendo dalla scelta del framework e del simulation tool, fino ad arrivare all'implementazione vera e propria dell'algoritmo di routing per la rete FANET. L'intero sistema proposto è installato su Ubuntu 16.04 LTS perché offre una migliore compatibilità, sopratutto con il simulation tool scelto.

5.1 OMNeT++

OMNeT++ (omnetpp.org) è un framework di simulazione di rete ad eventi discreti, modulare ed orientato agli oggetti. Ha un'architettura generica, quindi può essere utilizzato per la modellazione e per la simulazione di qualsiasi sistema in cui questo approccio è adatto e può essere convenientemente mappato in entità che comunicano scambiando messaggi. Il software in sé non è un simulatore di nulla di concreto, ma piuttosto fornisce infrastrutture e strumenti per la scrittura di simulazioni. Una delle sue caratteristiche fondamentali è l'architettura dei componenti per i modelli di simulazione. Essi sono assemblati da componenti chiamati moduli che, se ben scritti, sono veramente riutilizzabili e possono essere combinati in vari modi come i blocchi LEGO.

I moduli possono essere collegati tra loro tramite porte (le cosiddette "gates") e combinati per formarne di composti. La profondità dell'annidamento non è limitata. Essi comunicano attraverso il passaggio di messaggi che possono trasportare strutture dati arbitrarie. Inoltre, possono passare messaggi lungo percorsi predefiniti tramite porte e connessioni, o direttamente alla loro destinazione. Quest'ultimo caso è utile per simulazioni wireless, ad esempio. Possono anche avere parametri che vengono utilizzati per personalizzarne il comportamento e per parametrizzare la topologia del modello. I moduli al livello più basso della gerarchia sono chiamati "simple modules" e incapsulano il comportamento del modello. Essi sono programmati in C ++ e fanno uso della libreria di simulazione predefinita, fornita con il tool. Le simulazioni OMNeT++ possono essere eseguite con varie interfacce utente. Quelle grafiche e animate sono molto utili per scopi dimostrativi e di debug, mentre quelle a riga di comando sono le migliori per l'esecuzione batch.

La scelta è ricaduta su questo framework perché rappresenta una delle più valide opzioni fornite gratuitamente per uso accademico nell'ambito dei software simulatori di reti. La sua architettura si presta perfettamente al progetto, grazie alla struttura costruita sullo scambio di messaggi e alla possibilità di inviarne direttamente a destinazione, caratteristica utile per la

simulazione di ambienti wireless.

5.2 Scelta del Simulation Tool

Nella seconda fase dell'implementazione del progetto è necessario sceglierle un tool simulativo che permetta di modellare un collegamento Side-Link Mode 4 per la comunicazione U2U all'interno del UAV Swarm Network. La prima scelta è ricaduta sul Software SimuLTE (simulte.com) che è stato sviluppato appositamente per simulare collegamenti wireless con tecnologia LTE.

Il software presenta una serie di moduli che insieme formano la scheda di rete LTE. Essi permettono di creare collegamenti e di scambiare messaggi tra nodi della rete provvisti di tale interfaccia. SimuLTE però non presenta la gestione autonoma del Device-2-Device, quindi la Mode4, bensì supporta solamente le modalità precedenti che necessitano di una eNodeB che si occupi della parte di scheduling e routing.

Per questo motivo è stato scelto OpenCV2X (cs.ucc.ie/cv2x), un progetto basato sul precedente che introduce una NIC compatibile con la Mode 4 e fornisce un applicazione che permette di scambiare messaggi tra dispositivi vicini. Questo software però è stato progettato per lavorare senza l'utilizzo di IPv4, fattore che limita la simulazione e la allontana da un possibile scenario reale, dove l'utilizzo dello stack IP è quasi sempre necessario.

L'ambiente simulativo lavorerà quindi con la NIC di OpenCV2X, senza utilizzare il protocollo IP e con l'applicazione fornita dal tool, modificata per poter permettere lo scambio di messaggi tra UAV che non sono nel raggio di copertura reciproca. Questo tool presenta anche un'incompatibilità con "OpenSceneGraph" e "osgEarth", due plugins di OMNeT++ che insieme permettono di inserire mappe stradali all'interno dello scenario della simulazione, togliendo la possibilità di rappresentare aree geografiche, con tutte le loro caratteristiche.

5.3 OpenCV2X

OpenCV2X è un'implementazione open-source dello standard 3GPP Rel. 14. Si basa, come detto in precedenza, su una versione estesa di SimuLTE ed è pensato per funzionare con OMNeT++. Il software consente simulazioni di rete LTE Side-Link Mode 4. La Release 14 di 3GPP utilizza una nuova interfaccia radio Side-Link PC5 per la comunicazione V2V diretta grazie a LTE ProSe Device to Device (D2D) Versione 12 standard. Inoltre, specifica anche due nuove modalità di comunicazione, tra cui la Mode 4.

In modalità 4 (autonoma) ciascun device seleziona le proprie risorse radio per le comunicazioni D2D utilizzando un algoritmo di pianificazione distribuito, il SB-SPS. Pertanto, in Modalità 4 i veicoli possono funzionare senza infrastrutture, analogamente alle reti wireless ad-hoc. Questa è considerata la modalità di base poiché le applicazioni di sicurezza non possono dipendere dalla disponibilità di copertura cellulare.

L'obbiettivo di OpenCV2X è quello di fornire pieno supporto allo sviluppo di simulazioni veicolari che utilizzano questo tipo di tecnologia. Il framework fornisce una scheda di rete completa ed ottimizzata per lavorare con LTE Side-Link e un'applicazione base da cui derivarne diverse, a seconda di ciò che si vuole simulare. Nella scheda di rete sono stati sviluppati i livelli fisico e MAC, in modo da ricreare completamente tutto lo stack della comunicazione. Un'importante assunzione pero é necessaria, siccome questa scheda di rete LTE non é basata sullo stack IP, non é compatibile con applicazioni che necessitano della configurazione IPv4. Perciò, l'intero sistema simulato non utilizzerà lo stack TCP/IP.

Nel progetto di simulazione sono utilizzati entrambi i moduli forniti, con appropriate modifiche che verranno descritte nella sezione seguente. Anche se la quarta generazione delle tecnologie per le reti mobili non è l'ultima disponibile, OpenCV2X si pone come il precursore di un framework che può essere esteso al 5G, con potenziali casi d'uso veicolari ulteriormente ampliati nella Release 15 del 3GPP.

5.4 Scenario

Lo scenario base di OpenCV2X presenta un alcuni moduli che non sono necessari al progetto di simulazione. Al suo interno si trovano porzioni di codice che servono per l'integrazione di Sumo (simulatore di traffico veicolare urbano) e Veins (framework per l'esecuzione di simulazioni veicolari), che sono state eliminate in modo da mantenere soltanto i moduli utili alla comunicazione. In Fig. 5.1 è illustrato l'intero scenario simulativo.

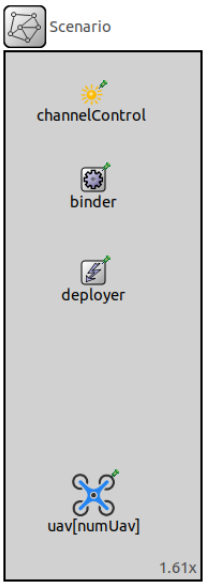

Figura 5.1: Scenario del progetto di simulazione

I primi tre componenti dall'alto sono specifici di SimuLTE, quindi anche di OpenCV2X e servono per la gestione delle comunicazioni attraverso la tecnologia LTE, mentre il primo dal basso (sempre in riferimento alla Fig. 5.1) rappresenta la catena variabile di UAV, fulcro del progetto.

Il Binder è un concetto importante in SimuLTE utilizzato principalmente per gestire gli UE nella simulazione. Vi si accede dai dispositivi quando si vuole determinare l'interferenza. Il Deployer invece, gestisce il dispiegamento dei nodi e mantiene la loro posizione nella simulazione.

I device sono direttamente derivati da quelli del framework, dai quali sono stati eliminati i moduli per l'integrazione con Sumo e Veins. Al loro interno troviamo quindi la parte per la gestione della mobilità, la scheda di rete e lo slot per il caricamento di una specifica applicazione.

Lo scambio di messaggi è interamente gestito da OMNeT++, attraverso connessioni dirette tra i dispositivi, mentre la gestione di essi è svolta dalla scheda di rete che si interfaccia con l'applicazione base fornita dal framework. L'applicazione è stata modificata in modo tale da gestire il routing dei messaggi verso il destinatario e salvare le statistiche di invio/ricezione.

5.5 LteNic

Siccome non è presente una cella (eNodeB), il livello MAC deve gestire la pianificazione dei pacchetti sulla base dell'algoritmo SB-SPS e inoltrarli attraverso il livello fisico (PHY). Tutto ciò è implementato nel modulo "LteMacVueMode4" di (15), che modifica "LteMacUeD2D" fornito da SimuLTE, aggiungendo la logica per la generazione di richieste di scheduling direttamente al livello MAC. Ciò significa che "LteMacVueMode4" non dipende più da una eNodeB per la scelta dei TBs.

Come mostrato in Fig. 5.2, dopo aver ricevuto un messaggio "newDataPkt" dal livello RLC, il livello MAC innesca la generazione di uno "SchedulingGrant", che comporta la determinazione delle risorse in base ad alcuni parametri preconfigurati correlati al livello PHY. Nel caso in cui uno "SchedulingGrant" è già disponibile viene programmato direttamente l'invio del pacchetto ricevuto. Quando un TB non è disponibile al momento della trasmissione, il livello MAC invia una richiesta SCI senza il MAC PDU al livello PHY. Il livello PHY trasmetterà in broadcast uno SCI senza TB associato.

Il livello PHY comprende la maggior parte dell'implementazione dell'algoritmo SB-SPS. "LtePhyVueMode4" è basato su "LtePhyUeD2D" di SimuLTE, che è stato esteso per implementare l'algoritmo di scheduling. Il modulo include la "Sensing Window" che mantiene la storia di SCIs e TBs e deter-

mina i CSRs ("Candidate Single-Subframe Resources"). Inoltre, si occupa della registrazione e della segnalazione del parametro "Channel Busy Ratio" (CBR) che misura la congestione del canale. Nel livello PHY vengono invece implementati i messaggi SCI insieme ai mezzi per generarli e decodificarli.

In particolare, per conformarsi alle linee guida della simulazione 3GPP (16), il modello di canale "WINNER + B1 LOS" e il "Nakagami Fading Model" sono stati implementati all'interno di SimuLTE.

La comunicazione tra i livelli MAC e PHY durante l'esecuzione dell'algoritmo SB-SPS è composta da:

- **CBR reporting:** il livello PHY riporta CBR dopo ogni sub-frame. Ciò è mostrato in Fig. 5.2 come "CBR report message".
- **CSR requests:** il livello MAC richiede un set di CSRs dal livello PHY e dopo il filtraggio, il PHY restituisce il 20% dei CSRs, basandosi sul calcolo del RSSI.
- **Data request:** il livello MAC invia il TB al livello PHY oppure il livello PHY invia un pacchetto ricevuto al livello MAC.
- **SCI request:** il livello MAC può richiedere al livello PHY di inviare un messaggio SCI con lo "SchedulingGrant" configurato al momento.

5.6 Mobility

Come descritto nel capitolo precedente è stata scelta una mobilità di tipo statico, che impone agli UAV di assumere la posizione iniziale e mantenerla fino alla fine della simulazione. La particolarità di questa Mobility è che permette ad N device di posizionarsi a distanza di 800m l'uno dall'altro, in modo del tutto parametrico. Il modulo base utilizzato per gli UAV ha una sezione apposita in cui impostare la mobilità scelta, definita come da libreria di INET.

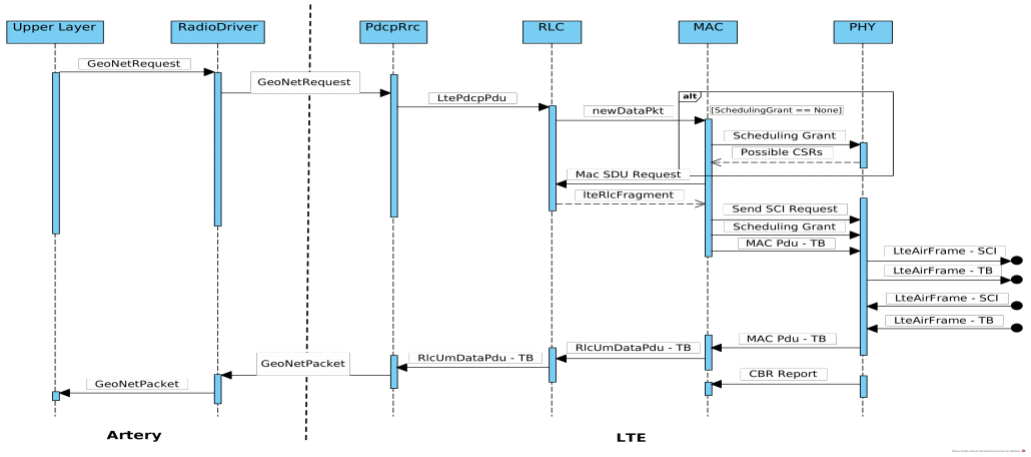

Figura 5.2: Diagramma di sequenza dello scambio di messaggi in OpenCV2X

La Mobility utilizzata è una versione modificata della "StaticGridMobility", fornita nel framework INET 3.6.6. Il codice sorgente originale è mostrato in Fig. 5.3. Nella versione standard tutti gli host vengono posizionati in una griglia rettangolare. L'area utilizzabile (area vincolata meno i margini su ciascun lato) viene suddivisa in celle più piccole, dimensionate a seconda dei parametri in input. Gli host vengono posizionati al centro di ogni cella. Per impostazione predefinita, il numero di colonne e righe segue le proporzioni dell'area utilizzabile e le distanze tra le celle vengono calcolate in base alle dimensioni della griglia.

```
//
// Places all hosts in a rectangular grid.
// The usable area (constraint area minus margins on each side) is split into smaller cells
// (with separationX,separationY size). Hosts are placed in the middle of each cell.
// By default, the number of columns and rows follow the aspect ratio of the usable area.
// By default stepX and stepY are calculated based on the number of columns and rows.
//
simple StaticGridMobility extends MobilityBase
{
    parameters:
        double marginX @unit(m) = default(0m);
        double marginY @unit(m) = default(0m);
        int numHosts;
        int columns = default(int(ceil(sqrt(numHosts * (this.constraintAreaMaxX - this.constraintAreaMinX - 2 * marginX) / (this.constraintAreaMaxY - this.constraintAreaMinY - 2 * marginY)))));
        int rows = default(int((numHosts + columns - 1) / columns));
        double initialZ @unit(m) = default(0m);
        double separationX @unit(m) = default((this.constraintAreaMaxX - this.constraintAreaMinX - 2 * marginX) / columns);
        double separationY @unit(m) = default((this.constraintAreaMaxY - this.constraintAreaMinY - 2 * marginY) / rows);
        @class(StaticGridMobility);
}
```

Figura 5.3: Codice sorgente di StaticGridMobility.ned, INET 3.6.6

Per posizionare gli UAVs come deciso in fase di modellazione è necessario **cambiare le seguenti due righe:**

```
int columns = default(int(ceil(sqrt(numHosts *
    (this.constraintAreaMaxX
    - this.constraintAreaMinX
    - 2 * marginX) / (this.constraintAreaMaxY
    - this.constraintAreaMinY - 2 * marginY)))));
int rows =
    default(int(
    (numHosts + columns - 1) / columns));
```

In:

```
int columns = numHosts;
int rows = 1;
```

In questo modo, tutti i device vengono posizionati su un'unica riga, uno per colonna, andando a formare una catena che mantiene la distanza passata come parametro tra i vari UAVs.

5.7 Mode4App

L'applicazione fornita con il framework OpenCV2X è stata progettata dagli autori per funzionare in collaborazione con Sumo e Veins. Siccome non è necessario mantenere la cooperazione tra i vari software, sono state rimosse anche qui le parti necessarie per l'integrazione ed è stata mantenuta soltanto la parte di gestione dei messaggi in arrivo e creazione di quelli in uscita.

L'applicativo è formato da tre diverse funzioni principali:

- **Initialize**
- **HandleSelfMessage**
- **HandleLowerMessage**

La prima funzione serve per avviare l'applicazione e registrare il nodo della rete con il Binder. In questa fase vengono inizializzate tutte le variabili

che servono, vengono registrati i parametri impostati nel file di configurazione della simulazione e creati gli spazi per registrare le varie statistiche. La seconda e la terza funzione invece servono rispettivamente per gestire i messaggi che il nodo invia a se stesso e per quelli che provengono dallo strato inferiore dell'architettura di rete, ciò quelli che riceve da altri nodi.

Dopo aver inizializzato il programma, l'ultimo step della prima funzione si occupa di schedulare un timer che invia un SelfMessage ad ogni intervallo di tempo specificato nel file di configurazione attraverso la variabile temporale "period". Questo messaggio attiva la seconda funzione che crea un semplice messaggio di Alert e lo invia in broadcast. Nel momento in cui un altro nodo nel raggio di copertura riceve il messaggio in questione, si attiva la terza funzione che si occupa di registrare l'evento, con tutte le statistiche associate. I messaggi di Alert sono definiti nel file "AlertPackage.msg" e contengono semplicemente il timestamp della creazione del messaggio e il numero di sequenza.

5.8 Routing a livello applicazione

L'applicazione fornita da OpenCV2X è stata modificata per poter applicare un algoritmo di routing statico a livello applicazione. Per prima cosa è stato aggiunto il campo destinatario nel messaggi di Alert, in modo tale che ogni nodo della rete possa sapere al momento della ricezione di uno di essi, se deve emettere le statistiche o meno e in tal caso inoltralo. Il codice sorgente di "AlertPacket.msg" è illustrato nell'Appendice A. Come descritto in fase di modellazione vogliamo che solamente il primo nodo possa inviare dei messaggi, quindi bisogna impostare il period di default per ognuno a 0, in modo tale che non scheduli mai la creazione di un nuovo SelfMessage e imporre soltanto al primo nodo della catena un valore diverso. La funzione HandleSelfMessage è stata ampliata in modo tale da aggiungere al pacchetto che viene creato ogni volta un valore per il campo destinatario, preso direttamente dal file di configurazione. In questo modo è possibile imporre l'ultimo

nodo della catena come destinatario di tutti i messaggi creati dal primo.

Tutte queste modifiche sono necessarie per poter applicare l'algoritmo descritto in fase di modellazione. Il vero e proprio routing si svolge nella funzione HandleLowerMessage, che si occupa di gestire tutti i messaggi ricevuti da ogni nodo. Per prima cosa controlla se il pacchetto ricevuto è destinato a lui ed in caso positivo registra il tempo di ricezione, la dimensione del pacchetto ricevuto, incrementa il contatore ed elimina il messaggio. In caso contrario invece inoltra il messaggio in broadcast e salva il numero di sequenza, in modo tale da registrare l'avvenuta gestione di quel determinato pacchetto.

Quando un messaggio viene inoltrato da un UAV, essendo il sistema basato sulla comunicazione in broadcast, esso verrà ricevuto da entrami i suoi vicini, ma soltanto quello successivo dovrà inoltrarlo. Per questo motivo è stato aggiunto un ulteriore controllo sul numero di sequenza del pacchetto in modo tale da non inoltrarlo se quest'ultimo è già stato gestito. Il codice sorgente delle funzioni è riportato nell'Appendice A.

Capitolo 6

Validazione

In questo capitolo è proposta una valutazione delle prestazioni dell'algoritmo proposto applicato della rete di droni simulata. Le metriche utilizzate sono quelle standard per la misurazione della qualità della rete, quali PDR, Throughput e ritardo nella consegna dei pacchetti. L'intero sistema proposto è stato testato su Ubuntu 16.04 LTS, utilizzando Cmdenv per l'esecuzione delle simulazioni in batch.

6.1 Configurazione della simulazione

Per poter eseguire una simulazione sulla rete di UAVs precedentemente modellata e descritta è necessario impostare alcuni parametri che riguardano la scheda di rete, il collegamento Side-Link, la Mobility, il mittente e il destinatario dei pacchetti generati. Tutte queste impostazioni sono disponibili nel file "omnetpp.ini", derivato sempre dal framework OpenCV2X. Tra i vari parametri legati alla LteNic c'è il "packet size", cioè la dimensione dei pacchetti che l'applicazione genera.

Altri parametri significativi sono legati allo Scenario, quali numero di UAV presenti nella rete, distanza orizzontale tra di essi e dimensioni dell'area di lavoro. Sempre in questa sezione è importante specificare il destinatario dei vari messaggi, che è stato impostato sull'ultimo UAV della catena e il period per il primo, in modo tale da attivare la generazione di SelfMessage ad ogni intervallo di tempo.

La simulazione è stata quindi impostata inizialmente per lavorare con 10 UAVs, per poi aumentare il numero fino a 100 a step di 10. Il carico di lavoro su una rete così ampia si è dimostrato molto elevato e poco gestibile da parte dell'algoritmo scelto e quindi è stato modificato questo parametro. La nuova configurazione prevede inizialmente 2 UAVs, poi 4, 6, 8, 10, 15, 20 ed infine 25. Una rete di dimensioni maggiori non riesce raggiungere prestazioni accettabili. La durata dei test è stata impostata a 70 secondi, ma i primi 10 vengono scartati per dare tempo al sistema di stabilizzarsi.

Gli altri due parametri fondamentali per la configurazione sono il "packet size" ed il "period" che sono stati impostati rispettivamente a 235 Bytes e 0.1s. Questo perché, dopo alcuni test, è stato verificato che la LteNic fornita da OpenCV2X non implementa la frammentazione dei pacchetti e la dimensione massima raggiungibile è appunto questa. Per quanto riguarda il period invece è stato scelto 0.1s perché, come si vedrà nei test successivi, è il valore più basso che non intacca pesantemente le prestazioni dell'intero sistema.

Di seguito, nella Tab. 6.1, sono riassunti i parametri principali della

simulazione.

Parametro	Valore
Simulation Time	70s
NumUAVs	2,4,6,8,10..25 step 5
SeparationX	800m
Period UAV[0]	0.1s
DestHost	last
PacketSize	235 bytes

Tabella 6.1: Riassunto dei parametri della simulazione

6.2 Scelta delle metriche

Per valutare a pieno le prestazioni dell'algoritmo proposto sono state scelte alcune metriche, tipiche delle comunicazioni in rete.

- **Packet Delivery Ratio (PDR)**
- **Delay o ritardo medio nella ricezione dei pacchetti**
- **Throughput**

Per poter registrare tutte le statistiche necessarie sono state aggiunte alcune variabili che si occupano di memorizzare il ritardo di ogni pacchetto ricevuto, la dimensione e il numero totale dei messaggi. Queste variabili sono tipiche di OMNeT++ e vengono gestite attraverso delle funzioni "emit" che permettono di emettere e salvare le statistiche su file esterni e di creare dei dataset, in modo tale da poter poi esportare dei grafici.

Queste metriche sono state scelte perché insieme danno una visione generale del funzionamento dell'algoritmo, in quanto permettono di capire l'efficienza della rete in termini di rapporto tra messaggi inviati e ricevuti, il ritardo nella consegna e la quantità di dati scambiati all'interno della finestra di tempo selezionata.

6.3 Risultati finali

Le simulazioni con il numero di UAVs variabile e con period e packet size fissi a 0.1s e 235 bytes, sono state eseguite utilizzando il tool a riga di comanda Cmdenv. Questo perché è direttamente disponibile in OMNeT++ e permette di eseguire le simulazioni in batch, variando automaticamente i parametri configurati nel file omnetpp.ini. Come detto in precedenza, l'unico parametro che può variare è il numero di UAVs, che verrà indicato con N. Sono state quindi valutate le prestazioni dell'algoritmo in termine di **PDR** (Fig. 6.1), **Delay** (Fig. 6.2) e **Throughput** (Fig. 6.3).

6.3.1 PDR

Il valore del PDR è stato calcolato sulla base dei messaggi inviati dal primo UAV della catena nella finestra di tempo analizzata, confrontando il valore registrato con quello dei messaggi ricevuti dall'ultimo device. Si può notare nel grafico in Fig. 6.1 come l'andamento del suo valore all'aumentare del numero di UAVs sia leggermente decrescente, indice di una piccola congestione della rete che va via via sempre ad aumentare. Si può affermare che il sistema gode di un buon funzionamento fino a quando l'indice non scende sotto lo 0.8-0,75, cioè circa il 75-80%. Ciò significa che l'algoritmo in questione può funzionare bene in una piccola rete, con un numero esiguo di UAVs, perché l'andamento del grafico evidenzia la discesa lineare dell'indice che oltre i 20 UAVs scende sotto il 75%.

6.3.2 Delay

Il delay viene calcolato in base al timestamp registrato nel momento in cui il pacchetto viene inviato dal primo UAV, confrontando il valore con il tempo della simulazione al momento della ricezione da parte dell'ultimo device della catena. Ogni pacchetto subisce dei ritardi diversi che possono essere influenzati da alcuni fattori che creano interferenza al segnale radio. Dal grafico in Fig. 6.2 si può notare che il sistema registra un ritardo medio inferiore ai 0.5s

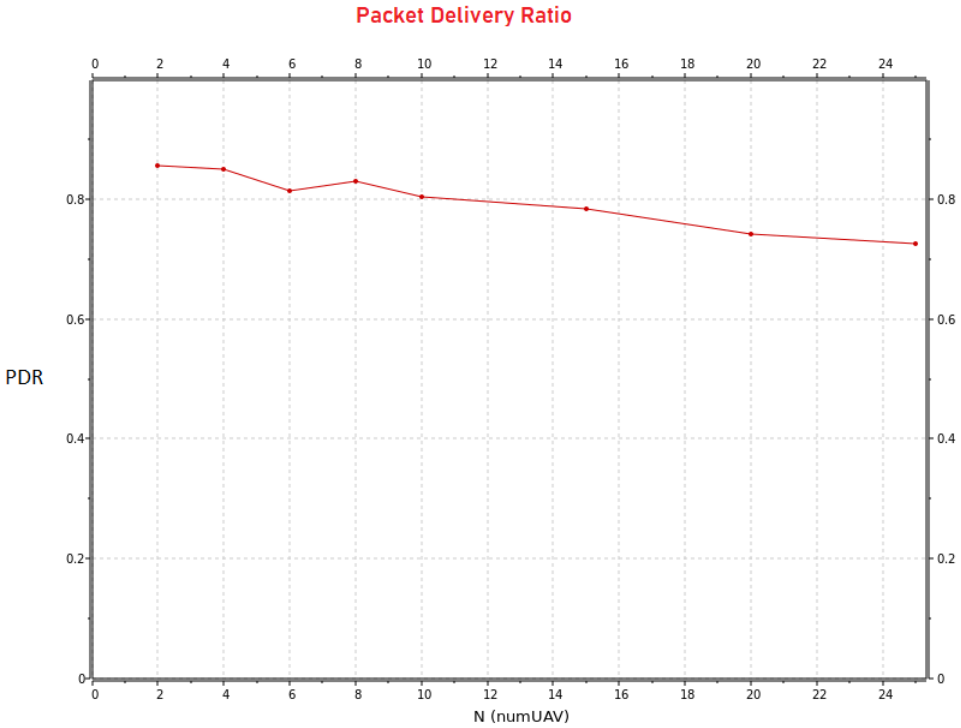

Figura 6.1: Packet Delivery Ratio del sistema simulato al variare del numero degli UAVs

quando il numero di UAVs è inferiore a 10, per poi incrementare linearmente con il numero dei device. Oltre i 10 UAVs il ritardo diventa molto alto ed sarebbe impossibile utilizzare questo algoritmo per applicazioni real-time. Ciò significa che l'algoritmo riesce ad ottenere discreti risultati quando la rete è piuttosto ridotta.

6.3.3 Throughput

Per monitorare la quantità di dati invitati dal mittente al ricevente, viene utilizzata una variabile che registra e somma la dimensione di tutti i pacchetti ricevuti dall'ultimo UAV. In questo modo vengono conteggiati tutti i bytes scambiati durante la finestra di tempo di 60s in cui vengono registrate le statistiche durante la simulazione. Siccome i pacchetti scambiati sono tutti

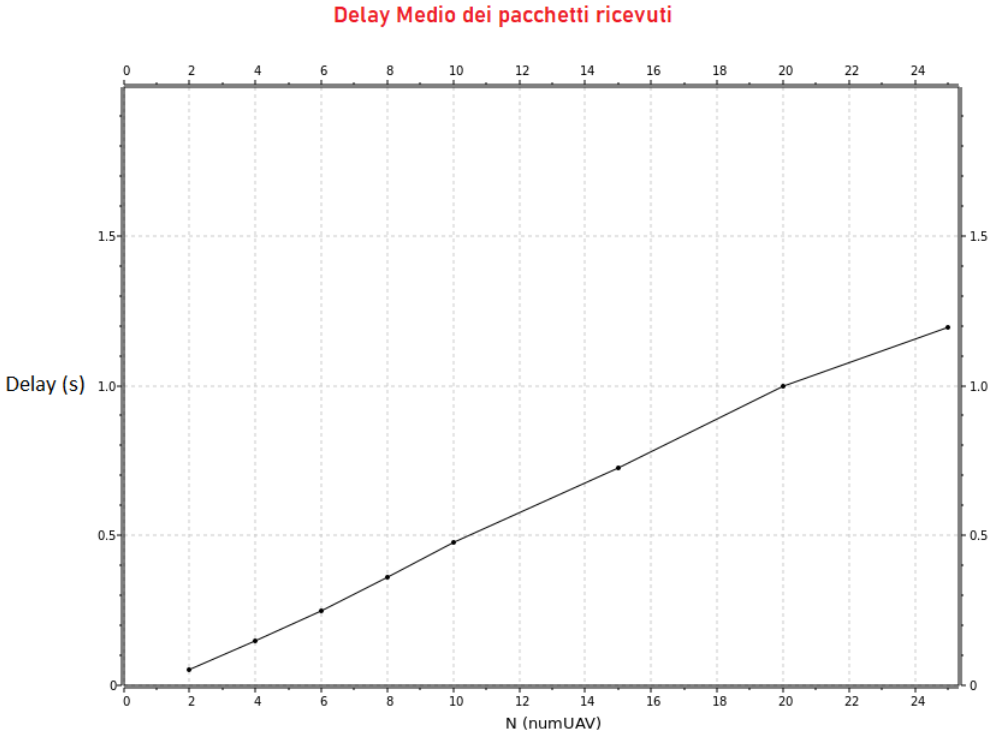

Figura 6.2: Delay medio nella consegna dei pacchetti del sistema simulato al variare del numero degli UAVs

da 235 bytes e non è implementata la frammentazione, il throughput dipende esclusivamente dal numero di pacchetti ricevuti dal destinatario. Ciò significa che al variare del numero di UAVs non cambia il valore dei pacchetti creati, ma soltanto il numeri di hop che essi devono compiere per giungere a destinazione. Come si può notare nel grafico in Fig. 6.3 il picco si ha con il minor numero di UAVs possibile, cioè 2. L'andamento però è abbastanza lineare e non ci sono cali bruschi all'aumentare dei device, questo perché il ritardo nella consegna di ogni pacchetto non è così elevato da non essere registrato nella simulazione. Il throughput si attesta quindi su circa 2000-2500 byte/s.

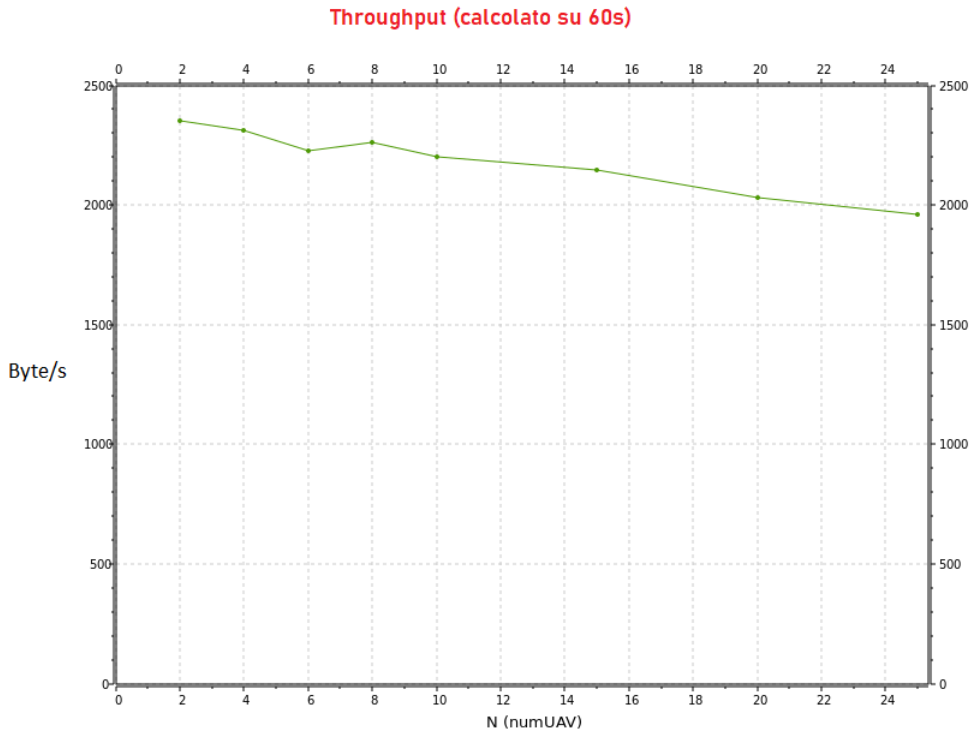

Figura 6.3: Throughput del sistema simulato al variare del numero degli UAVs, calcolato in una finestra di 60s

6.4 PDR al variare dell'intervallo di tempo di trasmissione

Dopo aver compiuto dei test con numero di UAVs variabile e valutato le performance dell'algoritmo sulla base di un intervallo di tempo di trasmissione di 0.1s, è interessante capire come varia l'indice PDR a seconda di quanti pacchetti vengono inviati in un determinato periodo di tempo. Per far ciò è necessario fissare il numero di UAVs e modificare il period, per poi confrontare il numero di messaggi ricevuti con quello degli inviati. Dai test precedenti si può notare che il sistema funziona benissimo con un numero basso di UAVs, per poi subire un degrado delle prestazioni man mano che il numero dei device sale. Un livello di funzionamento buono si ottiene con 10

UAVs, perché il PDR non scende sotto lo 0.8, il delay medio è stabile sotto 0.5s e il throughput è circa 2300 byte/s.

Nel grafico in Fig. 6.4 si può notare che con valori più bassi di 0.1s il PDR del sistema cala bruscamente intorno al 40%, questo avviene perché c'è una forte congestione nella rete e solo pochi pacchetti riescono ad arrivare a destinazione. Il valore invece si attesta su circa l'80%, quando l'intervallo di tempo di trasmissione sale. Ciò significa che 0.1s è il valore che permette di raggiunge il miglior rapporto tra il numero di pacchetti inviati e quelli ricevuti.

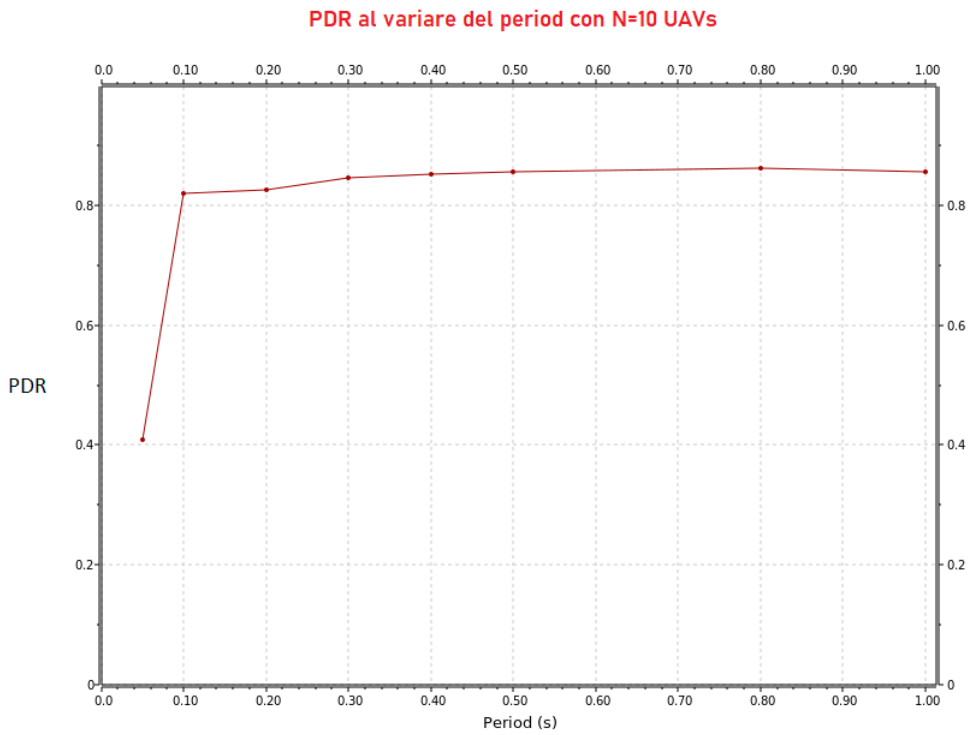

Figura 6.4: PDR del sistema simulato con N=10 UAVs al variare del period

Conclusioni

Lo scopo principale di questa serie di simulazioni è quello di testare le performance del Side-Link esteso per la comunicazione in una rete di UAVs multi-hop. L'algoritmo proposto a livello applicazione permette la consegna dei pacchetti al destinatario indipendentemente dal numero di hop necessari per raggiungerlo. Con questo tipo di estensione é quindi possibile scambiare messaggi con UAVs che non sono nel raggio di comunicazione della scheda di rete del nodo in questione. L'estensione fatta a Side-Link si é dimostrata quindi efficace e funzionate.

In generale, le performance misurate in fase di valutazione evidenziano un funzionamento del sistema simulato che registra prestazioni che calano linearmente con l'aumento dei nodi, questo perché un pacchetto impiega piú tempo ad arrivare a destinazione in relazione al numero di hop che deve compiere. Ciò significa che questo tipo di soluzione è utile per applicazioni ridotte e che richiedono poche risorse, sia in termini di comunicazione, sia in termini di unità computazionali presenti nella rete. Una rete troppo numerosa porterebbe a performance non adatte ad una comunicazione real-time.

Per rendere questo ambiente di simulazione più simile possibile ad uno scenario reale, nei lavori futuri si potrebbe aggiungere in primis la compatibilità con lo stack IPv4, per rendere la LteNic compatibile con più applicazioni possibili. Si potrebbe poi aggiungere una mobilità dinamica, ma ciò richiederebbe la sostituzione dell'algoritmo testato con uno più evoluto, che tenga in considerazione anche le nuove posizioni dei vari componenti della rete. Infine, per completare l'intero lavoro dovrebbero essere aggiunte al modu-

lo creato per la rappresentazione di un UAV, tutte le componenti che ogni device presenta realmente, compresa la batteria. In questo caso sarebbe necessario aggiungere una parte che si occupi della gestione dell'alimentazione, argomento molto ampio e ancora in fase di ricerca da parte dei vari esperti del settore.

Appendice A

Codice Sorgente

A.1 Alert Packet.msg

```
packet AlertPacket
{
    unsigned int sno;
    simtime_t timestamp;
    unsigned int dest; //destination of the pkt
}
```

A.2 Initialize

```
void Mode4App::initialize(int stage)
{
    Mode4BaseApp::initialize(stage);
    if (stage==inet::INITSTAGE_LOCAL){
        // Get the binder
        binder_ = getBinder();
        // Get our UE
        cModule *ue = getParentModule();
```

```
        //Register with the binder
        nodeId_ = binder_->registerNode(ue, UE, 0);
        // Register the nodeId_ with the binder.
        binder_->setMacNodeId(nodeId_, nodeId_);
    } else {
        if (stage==inet::INITSTAGE_APPLICATION_LAYER){
            EV << "INIZIO" << endl;
            selfSender_ = NULL;
            nextSno_ = 1;
            last_forwarded_ = 0;
            selfSender_ = new cMessage("selfSender");
            size_ = par("packetSize");
            period_ = par("period");
            priority_ = par("priority");
            duration_ = par("duration");
            cbr_ = registerSignal("cbr");
            sentMsg_ = registerSignal("sentMsg");
            delay_ = registerSignal("delay");
            rcvdMsg_ = registerSignal("rcvdMsg");
            tput_ = registerSignal("tput");
            double delay = 0.001*intuniform(0,1000,0);
            if(period_>0){
                scheduleAt(
                (simTime() + delay).trunc(SIMTIME_MS),
                selfSender_);
            }
        }
    }
}
```

A.3 HandleLowerMessage

```
void Mode4App::handleLowerMessage(cMessage* msg)
{
    if (msg->isName("CBR")) {
        Cbr* cbrPkt = check_and_cast<Cbr*>(msg);
        double channel_load = cbrPkt->getCbr();
        emit(cbr_, channel_load);
        delete cbrPkt;
    } else {
        AlertPacket* pkt =
            check_and_cast<AlertPacket*>(msg);
        if (pkt == 0)
            throw
            cRuntimeError(
            "Mode4App::handleMessage - FATAL!"
            +"Error when casting to AlertPacket");
        EV << "getDest: " << pkt->getDest() <<
            " currentNode: " <<
            this->getParentModule()->getIndex() <<
            endl;
        FlowControlInfoNonIp *source =
            check_and_cast<FlowControlInfoNonIp*>
                (msg->getControlInfo());
        if(pkt->getDest()==
        this->getParentModule()->getIndex()){
            simtime_t delay =
                simTime() - pkt->getTimestamp();
            if(simTime()>10){
                emit(delay_, delay);
                emit(rcvdMsg_, (long)1);
```

```
            emit(tput_, (long)size_);
        }
        EV << "PACCHETTO ARRIVATO" << endl;
        EV << "Mode4App::handleMessage - "
        +"Packet received: SeqNo[" <<
        pkt->getSno() <<
        "] Delay[" << delay << "]" << endl;
        delete msg;
    }else{
        if(pkt->getDest()>
        this->getParentModule()->getIndex()){
            EV << "getSno: " <<
            pkt->getSno() << " lastForwarded: " <<
            last_forwarded_ << endl;
            if(pkt->getSno()>last_forwarded_){
                last_forwarded_=pkt->getSno();
                EV << "INOLTRO" << endl;
                pkt->removeControlInfo();
                auto lteControlInfo =
                    new FlowControlInfoNonIp();
                lteControlInfo
                    ->setSrcAddr(nodeId_);
                lteControlInfo
                    ->setDirection(D2D_MULTI);
                lteControlInfo
                    ->setPriority(priority_);
                lteControlInfo
                    ->setDuration(duration_);
                lteControlInfo
                    ->setCreationTime(simTime());
                pkt
```

```
                    ->setControlInfo(lteControlInfo);
                    Mode4BaseApp::
                        sendLowerPackets(pkt);
                }else{
                    EV <<
                    "NON_INOLTRO_(se_no_creo_loop)" <<
                    endl;
                }
            }else{
                EV << "NON_INOLTRO" << endl;
            }
        }
    }
}
```

A.4 HandleSelfMessage

```
void Mode4App::handleSelfMessage(cMessage* msg)
{
    if (!strcmp(msg->getName(), "selfSender")){
        AlertPacket* packet = new AlertPacket("Alert");
        packet->setTimestamp(simTime());
        packet->setByteLength(size_);
        packet->setSno(nextSno_);
        int destHost = par("destHost");
        destHost = destHost - 1;
        packet->setDest(destHost);
        nextSno_++;
        last_forwarded_=nextSno_;
        auto lteControlInfo =
            new FlowControlInfoNonIp();
```

```
            lteControlInfo->setSrcAddr(nodeId_);
            lteControlInfo->setDirection(D2D_MULTI);
            lteControlInfo->setPriority(priority_);
            lteControlInfo->setDuration(duration_);
            lteControlInfo->setCreationTime(simTime());
            packet->setControlInfo(lteControlInfo);
            Mode4BaseApp::sendLowerPackets(packet);
            emit(sentMsg_, (long)1);
            scheduleAt(simTime() + period_, selfSender_);
        }
        else
            throw cRuntimeError("Mode4App::handleMessage"
            + "- Unrecognized self message");
    }
```

Bibliografia

[1] M. A. Khan, I. M. Qureshi, and F. Khanzada, "A hybrid communication scheme for efficient and low-cost deployment of future flying ad-hoc network (fanet)," *Drones*, vol. 3, p. 16, Feb 2019.

[2] İlker Bekmezci, O. K. Sahingoz, and Şamil Temel, "Flying ad-hoc networks (fanets): A survey," *Ad Hoc Networks*, vol. 11, no. 3, pp. 1254 – 1270, 2013.

[3] M. Mozaffari, W. Saad, M. Bennis, Y. Nam, and M. Debbah, "A tutorial on uavs for wireless networks: Applications, challenges, and open problems," *IEEE Communications Surveys Tutorials*, vol. 21, no. 3, pp. 2334–2360, 2019.

[4] K. Ganesan, J. Lohr, P. B. Mallick, A. Kunz, and R. Kuchibhotla, "Nr sidelink design overview for advanced v2x service," *IEEE Internet of Things Magazine*, vol. 3, no. 1, pp. 26–30, 2020.

[5] M. Noor-A-Rahim, G. G. M. N. Ali, Y. L. Guan, B. Ayalew, P. H. J. Chong, and D. Pesch, "Broadcast performance analysis and improvements of the lte-v2v autonomous mode at road intersection," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 10, pp. 9359–9369, 2019.

[6] R. Fritzsche and A. Festag, "Location-based scheduling for cellular v2v systems in highway scenarios," in *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*, pp. 1–5, 2018.

[7] S. Lien, D. Deng, C. Lin, H. Tsai, T. Chen, C. Guo, and S. Cheng, "3gpp nr sidelink transmissions toward 5g v2x," *IEEE Access*, vol. 8, pp. 35368–35382, 2020.

[8] N. Bonjorn, F. Foukalas, and P. Pop, "Enhanced 5g v2x services using sidelink device-to-device communications," in *2018 17th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*, pp. 1–7, 2018.

[9] R. Molina-Masegosa and J. Gozalvez, "Lte-v for sidelink 5g v2x vehicular communications: A new 5g technology for short-range vehicle-to-everything communications," *IEEE Vehicular Technology Magazine*, vol. 12, no. 4, pp. 30–39, 2017.

[10] Y. Jeon and H. Kim, "An explicit reservation-augmented resource allocation scheme for c-v2x sidelink mode 4," *IEEE Access*, vol. 8, pp. 147241–147255, 2020.

[11] P. J. Honnaiah, N. Maturo, and S. Chatzinotas, "Foreseeing semi-persistent scheduling in mode-4 for 5g enhanced v2x communication," in *2020 IEEE 17th Annual Consumer Communications Networking Conference (CCNC)*, pp. 1–2, 2020.

[12] S. Sabeeh, P. Sroka, and K. Wesołowski, "Estimation and reservation for autonomous resource selection in c-v2x mode 4," in *2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pp. 1–6, 2019.

[13] L. Lin, Q. Sun, S. Wang, and F. Yang, "A geographic mobility prediction routing protocol for ad hoc uav network," in *2012 IEEE Globecom Workshops*, pp. 1597–1602, 2012.

[14] F. Wang, Z. Chen, J. Zhang, C. Zhou, and W. Yue, "Greedy forwarding and limited flooding based routing protocol for uav flying ad-hoc

networks," in *2019 IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, pp. 1–4, 2019.

[15] B. McCarthy and A. O'Driscoll, "Opencv2x mode 4: A simulation extension for cellular vehicular communication networks," in *2019 IEEE 24th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pp. 1–6, 2019.

[16] P. M. J. Meredith, "Study on lte-based v2x services (release 14)," tech. rep., technical specification. Technical report, 2016.

Ringraziamenti

Vorrei dedicare questo spazio a chi, con dedizione e pazienza, ha contribuito alla realizzazione di questo elaborato.

Un ringraziamento particolare va al mio relatore Marco Di Felice che mi ha seguito nei vari step della realizzazione dell'elaborato, fin dalla scelta dell'argomento. Grazie anche ai miei correlatori Enrico Natalizio e Angelo Trotta per i loro preziosi consigli e per avermi suggerito puntualmente le giuste modifiche da apportare al mio progetto di tesi.

Ringrazio infinitamente mia madre e mio padre, senza i loro insegnamenti e senza il loro supporto, questo lavoro non esisterebbe nemmeno. Grazie anche a tutta la mia famiglia, a partire da mia sorella fino alle mie cugine, tutti compresi, per il vostro supporto durante tutti gli anni di studio e per avermi fatto sempre sentire bravo in qualcosa, elogiando ciò che facevo.

Un immenso grazie va poi alla mia ragazza Chiara, oltre che per l'infinito affetto e l'immancabile supporto, per aver acceso in me, ormai due anni fa, quella scintilla di cui avevo bisogno per uscire dalla mia zona di comfort. Per me è stato un punto cruciale della mia crescita personale.

Ringrazio poi voi, tutte quelle persone che sono state la "Mia Bologna" in questi due anni magnifici. Edoardo, primo amico e coinquilino, grazie per tutti i caffè e per tutte le spremute, ah già e anche per il tuo supporto e per i momenti più indimenticabili di Via Matteotti. Giovanni, per l'esempio che sei per me, tu lo sai. Lorenzo, per aver condiviso gioie e dolori davanti al televisore. Simone ed Emiliano, per avermi accolto subito e trattato d'amico sin dal giorno zero, peccato che le nostre strade si siano divise così presto. I

ragazzi di via Mazzini 3, per quel pizzico di internazionalità che mancava nel mio carattere, mi avete sempre accolto e non mi sono mai sentito in difetto quando il mio inglese mi creava dei piccoli problemi. Lo Scarsenal, per tutti i momenti di svago che mi permettevano di staccare completamente la testa dalla mia quotidianità. Grazie anche a Viviana e Cristian, per tutto il tempo condiviso sui vari progetti a cui abbiamo collaborato.

Grazie a Filippo e a tutto gruppo della triennale, che mi sono sempre stati vicino, anche dopo il mio trasferimento.

Una menzione speciale poi va a Luigi, che mi ha accolto a Nancy fin da subito, senza avermi mai visto prima, in un momento storico dove per necessità si è costretti ad evitare persone estranee. Grazie per tutto quello che hai fatto per me mentre ero li.

Infine, grazie anche a voi, Michele, Sara e a tutti i miei amici che mi hanno sempre supportato, aiutato e ascoltato quando ne avevo bisogno durante tutta questa esperienza.