Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea Magistrale in Informatica

INTEGRAZIONE E VALIDAZIONE DI OMNET++ E GAZEBO PER SISTEMI MULTI-ROBOT MEDIANTE UN CASO D'USO

Relatore:
Chiar.mo Prof.
MARCO DI FELICE

Correlatore:
ANGELO TROTTA

Presentata da:
FEDERICO ZAPPI

Sessione III
Anno Accademico 2018/2019

Indice

1 Introduzione **1**

1.1 Il caso d'uso 3

1.2 Struttura dell'elaborato 4

2 Related Works **5**

2.1 Integrazione di ambienti diversi a scopo di una simulazione più realistica 5

2.1.1 Integrazione tra simulatori di traffico stradale e di rete 6

2.1.2 Integrazione tra simulatori di robot e di rete 11

2.2 Swarm intelligence per il movimento dei droni 14

2.2.1 Molle virtuali 15

2.2.2 Altri sistemi ispirati alla fisica 21

3 Architettura **23**

3.1 OMNeT++ 23

3.2 Gazebo 26

3.3 Unione dei due sistemi 28

3.3.1 L'alternanza nella simulazione 28

3.3.2 Gli elementi principali 29

3.3.3 Comunicazione e inizializzazione 31

3.3.4 Struttura dei messaggi 33

3.3.5 Lato Gazebo 34

3.3.6 Lato OMNeT++ 36

4 Caso di studio **41**
4.1 Introduzione 41
4.2 Molle dello sciame 43
4.2.1 Generazione e rimozione delle molle 44
4.3 Molle per evitare gli ostacoli 47
4.3.1 I sensori 47
4.3.2 Spostamento dovuto ai sensori 48
4.3.3 Molle sciame + Molle ostacoli 49
4.4 Realizzazione 49
4.4.1 Creazione e aggiornamento molle 50
4.5 OpenStreetMap 51

5 Valutazioni e Performance **53**
5.1 Scenario 53
5.1.1 Parametri della simulazione 55
5.2 Risultati 57
5.2.1 Valutazione del sistema (OMNeT++ & Gazebo) 57
5.2.2 Valutazione della mobilità dei droni 61

Conclusioni **81**

Bibliografia **83**

Elenco delle figure

2.1 Il legame del modello di mobilità con il simulatore di rete nel tempo 7
2.2 Architettura di TraNS 8
2.3 Sincronizzazione in Veins 9
2.4 Architettura di iTETRIS 10
2.5 Controllo di un UAV in HNMSim 11
2.6 Differenza nel rapporto tra simulatore di rete e simulatore del drone/robot 13
2.7 Diversi tipi di molle delle Repairing Units 16
2.8 Architettura e forze del sistema leader-followers con Turtlebots 18
2.9 Molle e risultati dei test del sistema leader-followers per il trasporto coordinato 19
2.10 Forze del sistema leader-followers per il tragitto con possibili ostacoli 20
2.11 Campi elettrici per il movimento dei droni 21

3.1 Simulazione in OMNeT++ dove un host invia all'altro un pacchetto 25
3.2 L'interfaccia grafica di Gazebo. 27
3.3 Schema di interazione tra OMNeT++ e Gazebo 28
3.4 Architettura del sistema integrato 30
3.5 La gestione dei sensori dei droni lato Gazebo 36
3.6 File .ned della simulazione 37

3.7 File .ned del modulo GazeboUAV 38
3.8 Ereditarietá dei moduli dei sensori 39

4.1 La forza che la molla virtuale esercita sui droni a seconda del Link Budget 43
4.2 Test dell'angolo acuto su drone A al momento della ricezione di un messaggio dal drone B 46
4.3 Esempio di formazione molle tra tre droni più o meno allineati 46
4.4 Gli 8 sensori dei droni 48
4.5 Schema di realizzazione del caso di studio 49
4.6 Mappa ricavata ad OpenStreetMap 52

5.1 Il modello della città utilizzato nelle simulazioni 55
5.2 Attesa di OMNeT++ variando il numero dei sensori 58
5.3 Attesa di OMNeT++ variando il numero di droni 59
5.4 Velocità di simulazione a confronto tra con e senza Gazebo . . 60
5.5 Velocità di simulazione a confronto tra con e senza Gazebo, variando il numero di droni 60
5.6 Droni formano una fila dietro al leader 61
5.7 Droni evitano ostacolo e imboccano una via 62
5.8 Coda di 2 droni affiancati presenta problemi a imboccare la via. 63
5.9 Drone rimane bloccato in un vicolo 63
5.10 Distacco dal leader (targetLinkBudget 20 dBm, k=2) 64
5.11 Distacco di leader con 2 follower (targetLinkBudget 20 dBm, k=5) 64
5.12 LB medio in città (senza svolte) al variare del numero di droni 65
5.13 LB medio in città (senza svolte) al variare del numero di droni (vel. leader 5 m/s) 66
5.14 Percorso con svolte, prima curva 67
5.15 Percorso con svolte, seconda curva problematica 68
5.16 Percorso con svolte, un drone si blocca nell'ultimo ostacolo . . 69
5.17 LB medio in spazio aperto al variare del numero di droni . . . 70

5.18 Coda nello sciame di 4 droni in spazio aperto 70
5.19 LB medio in spazio aperto al variare del numero di droni (vel. leader 5 m/s) 71
5.20 LB medio in spazio aperto al variare del numero di droni (targetLinkBudget 20 dBm) 71
5.21 Code dello sciame in spazio aperto (targetLinkBudget 20 dBm) 72
5.22 Percorso con ostacoli (foresta) 73
5.23 Distanza minima dagli ostacoli al variare del numero dei droni (vari test) 74

Capitolo 1

Introduzione

Negli ultimi anni sono diventati di maggior interesse sistemi multi-agente dove singoli robot a mobilitá autonoma comunicano tra di loro per coordinarsi e realizzare un certo tipo di servizio, ad esempio: l'esplorazione di un ambiente potenzialmente pericoloso, la ripresa video di un evento, la creazione di una rete di comunicazione a topologia dinamica. Per progettare e studiare questo tipo di sistemi si fa spesso affidamento a simulatori, che permettono di simulare il sistema in ambiente virtuale evitando cosí di dover effettuare test sul campo che potrebbero non essere facilmente realizzabili (per ragione di costi, sicurezza o regolamentazioni).
Esistono vari simulatori utilizzati in letteratura per lo studio di diversi tipi di sistemi. Tuttavia, nel modellare questo tipo di sistemi multi-agente, sono importanti sia gli aspetti di comunicazione che di controllo dei robot, e simulatori in grado di effettuare una simulazione accurata di entrambi gli aspetti sono ancora in fase di formazione.

Oggetto di questa tesi è stata la realizzazione di un sistema integrato per la simulazione di sistemi multi-agente dove sia la comunicazione che il controllo dei robot sono fattori importanti, e l'utilizzo di tale strumento per l'analisi di un caso d'uso, ovvero una forma di mobilità pensata per un gruppo di droni. Il sistema integrato è composto da due diversi simulatori, OMNeT++ [1] e Gazebo [2].

OMNeT++ è un simulatore ad eventi molto utilizzato per lo studio delle reti di comunicazione. Non è nato come simulatore di rete, e di base non comprende modelli per architetture e protocolli di rete, ma nel corso degli anni vari modelli sono stati creati a tale scopo, tra cui l'INET framework, la libreria mantenuta dal team di OMNeT++ e utilizzata come base per diversi progetti.

Gazebo è un simulatore 3D in grado di simulare accuratamente ed efficientemente gruppi di robot. Fornisce il supporto a 4 diversi physics engine, un'ampia selezione di sensori, una libreria di modelli di robot e elementi ambientali e la possibilità di estendere alcune sue funzionalità tramite la creazione di plugin. Dal 2009 permette l'integrazione con ROS (Robot Operating System [28]) e al momento il suo sviluppo è guidato dalla Open Source Robotic Foundation (OSRF).

L'unione dei due simulatori permette di sfruttare le caratteristiche di entrambi in una stessa simulazione, cosa desiderabile nello studio del comportamento dei droni: infatti, una funzionalità a noi utile di Gazebo che OMNeT++ può sfruttare è la gestione di vari tipi di sensori, come quello di prossimità per l'individuazione degli ostacoli (utilizzato nella tesi) o il dispositivo camera per la cattura delle immagini. Inoltre, permette all'utente di vedere la simulazione in ambiente 3D e di ricreare in esso gli scenari necessari; in questa tesi, ad esempio, si è utilizzato in Gazebo un ambiente di città ricavato usando come fonte dati OpenStreetMap per riprodurre un ambiente reale [29].

In pratica, Gazebo si occupa della parte di interazione con l'ambiente (scenario e sensori) mentre il resto della simulazione viene gestito da OMNeT++ (logica dei droni, mobilità e comunicazioni wireless).

Quello che alla fine avviene tra i due sistemi è un continuo alternarsi nella simulazione e passarsi i dati necessari: i dati dei sensori vengono generati da Gazebo e inviati a OMNeT++, mentre le posizioni dei droni sono decise e calcolate in OMNeT++ e vengono comunicate a Gazebo.

1.1 Il caso d'uso

Questa integrazione tra i due simulatori è stata pensata per lo studio del comportamento dei droni privi di pilota (UAV, Unmanned Aerial Vehicle), dispositivi volanti che possono essere pilotati in remoto da un umano, mediante un controller, o (il caso d'interesse) completamente lasciati liberi di muoversi in base a un programma eseguito su di essi (mobilità autonoma).

Inizialmente nati in ambito militare, i droni sono ora molto diffusi tra la popolazione mondiale, utilizzati anche semplicemente per svago da una persona qualunque. Ma ci sono molti scopi, molte idee che sono sorte negli ultimi anni riguardo ai possibili usi dei droni, sia per scopi commerciali che per pura utilità.

Tuttavia, l'utilizzo dei droni presenta un ostacolo: la regolamentazione del loro utilizzo, che non è pienamente definita. Essendo degli oggetti che si muovono nello spazio aereo, il loro possibile uso viene deciso principalmente dalle autorità che si occupano di questo ambiente, e possono esserci regole diverse a seconda dei paesi. Il fatto è che, per garantire che l'uso dei droni avvenga in sicurezza, è necessario esaminare con la dovuta cura la questione, e quindi i tempi per la decisione su come regolare i droni a mobilità autonoma e su dove permettere il loro utilizzo non sono brevi, e rallentano il momento della loro applicazione sul campo.

In ogni caso, prima di testare i droni nel mondo reale, è utile simulare il loro comportamento al computer.

In questa tesi, dopo aver integrato i simulatori, si è riprodotto un certo tipo di mobilità dei droni decentralizzata, ovvero la mobilità basata sulle molle virtuali presente in [15], dove un gruppo di droni si posiziona a una certa distanza dagli altri in modo da offrire una connessione wireless che garantisca una certa qualità del servizio richiesta.

Questa mobilità forma un gruppo compatto, uno sciame, che però rimane fermo in una zona. Usando sempre le molle, però, è possibile far muovere il gruppo di droni utilizzando un drone leader, che guida il gruppo verso una nuova zona.

A questo punto, però, il gruppo si muove, e in una zona con ostacoli i droni devono fare attenzione a non urtare nulla. Per fare questo vengono utilizzate delle molle che spingono il drone ad allontanarsi dall'ostacolo.

1.2 Struttura dell'elaborato

In seguito viene presentata la struttura del resto del documento, ovvero gli argomenti discussi nei successivi capitoli.

Nel Capitolo 2 vengono trattati alcuni articoli legati agli argomenti della tesi; si racconta certi loro aspetti e si indicano alcune differenze con il sistema realizzato. Nel Capitolo 3 viene presentata l'architettura del sistema integrato, preceduta da una descrizione dei due simulatori che lo compongono (OMNeT++ & Gazebo). In Capitolo 4 si descrive lo scenario che si è voluto realizzare tramite il sistema integrato e viene presentato nei dettagli il sistema delle molle utilizzato nella mobilià dei droni. Nel Capitolo 5 viene presentata l'analisi che si è condotta su sistema integrato e mobilità dei droni per valutarne il funzionamento. Nella valutazione dei droni, si è effettuato test in tre diversi scenari. Infine, nelle Conclusioni, sono presenti le considerazioni finali su quanto realizzato e i possibili sviluppi futuri proposti.

Capitolo 2

Related Works

In questo capitolo sono revisionati alcuni articoli che riguardano l'argomento della tesi, cioè l'unione di due diversi ambienti (simulatore + simulatore o simulatore + motore grafico) e sistemi di swarm intelligence per l'organizzazione del gruppo di droni nello spazio.

2.1 Integrazione di ambienti diversi a scopo di una simulazione più realistica

Il vantaggio di utilizzare dei simulatori è quello di poter studiare e progettare il comportamento dei soggetti di interesse senza dover testare ciò nella realtà, il che risparmia i preparativi di numerosi test per correggere i principali errori di progettazione e i possibili danni alle attrezzature o all'ambiente. Permette inoltre di testare un sistema che sarebbe altrimenti troppo costoso o troppo rischioso verificare nella realtà. Nel caso dei droni senza pilota, permette di effettuare test che altrimenti non si potrebbero realizzare al momento, a causa della regolamentazione in corso dei droni caratterizzati da questa particolare forma di mobilità.

Essendo utile e a volte necessario effettuare test tramite simulazione, risulta importante che il simulatore riproduca in maniera accurata il comportamento che il sistema da simulare mostrerebbe in un test reale, e questo

diventa un requisito importante del simulatore quanto più si è legati ad esso per la conduzione dei test (costi, rischi, regolamentazioni). Si presenta quindi un problema nella simulazione di uno sciame di UAVs, e in generale di gruppi di robot a mobilità autonoma che si affidano a sensori per la percezione dell'ambiente ma anche alla comunicazione con i loro vicini per la scelta delle decisioni da prendere, in quanto singoli strumenti che permettano di simulare accuratamente sia la parte di comunicazione che di controllo del robot non sono a disposizione. Alcuni lavori presenti in letteratura colmano questa lacuna proponendo come soluzione l'integrazione di strumenti/simulatori esistenti, singolarmente in grado di soddisfare un particolare requisito, che integrati insieme forniscono lo strumento desiderato.

Questo approccio permette di utilizzare strumenti di provata affidabilità, che hanno avuto tempo per correggere e migliorare il loro funzionamento e crescere in funzionalità e modelli disponibili. Inoltre, il fatto di utilizzare strumenti che già molte persone utilizzano e conoscono rende piú affidabile lo strumento e la simulazione realizzata con esso. Un lato negativo, tuttavia, è il lavoro di integrazione, che può essere anche problematico in quanto potrebbe essere necessario modificare il comportamento di un componente che si vorrebbe fosse gestito in altro modo (ad esempio definire certe caratteristiche della simulazione in un simulatore e doverle comunicare a un altro, che però ha un suo sistema che deve essere per forza utilizzato). Un altro svantaggio potrebbe consistere nelle performance del sistema così creato.

2.1.1 Integrazione tra simulatori di traffico stradale e di rete

I primi casi di integrazione tra un simulatore che gestisce gli spostamenti di un dispositivo mobile e un simulatore di reti di comunicazioni appaiono nell'ambito delle reti veicolari (VANET), dove si studia il traffico veicolare e la comunicazione tra i soli veicoli o tra loro e altri elementi (V2X). In [3] e in [5] si indica come, per valutare i protocolli sviluppati per le VANET, sia importante avere una comunicazione bidirezionale tra i due simulatori, in

quanto non solo la mobilità influenza le comunicazioni dei veicoli, ma anche i messaggi ricevuti da tali comunicazioni possono influenzare la mobilità del veicolo: si pensi ad esempio a un'applicazione che avvisa di un pericolo poco più avanti, e quindi della necessità di rallentare, oppure di una coda, che sarebbe opportuno evitare effettuando un cambio di percorso. La mobilità del veicolo, per essere realistica, non può quindi essere completamente nota prima della simulazione, come si faceva precedentemente [3], ma deve essere calcolata sul momento.

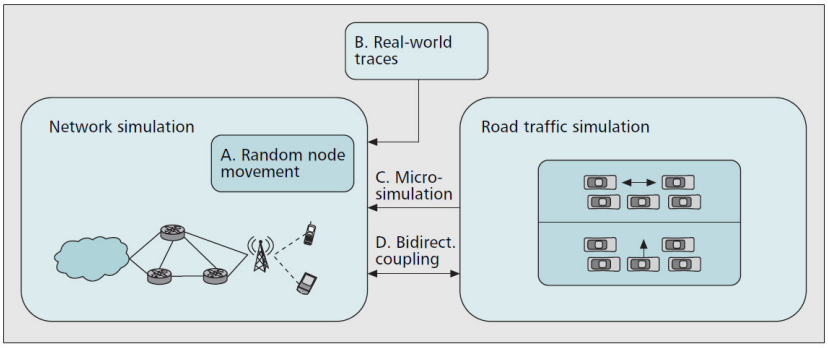


Figura 2.1: Il legame del modello di mobilità con il simulatore di rete nel tempo, da A verso D (immagine presa da [3]).

Esistono principalmente due modelli di mobilità veicolare: il modello macroscopico, che opera su flussi di veicoli, e il modello microscopico, che modella il comportamento di ogni singolo veicolo, e che è utilizzato per lo studio delle VANET. Per quanto riguarda i simulatori di traffico a livello microscopico, SUMO (Simulation for Urban MObility [4]) risulta il più completo, e negli anni è stato usato per vari progetti di ricerca. È open-source ed è stato ideato apposta per essere utilizzato dagli studiosi di questo campo di ricerca. Vari sistemi lo scelgono per integrarlo con i simulatori di rete.

Il sistema integrato TraNS (Traffic and Network Simulation Environment [5]) unisce SUMO con il simulatore di rete NS-2 [6]. Non è il primo caso di una integrazione tra simulatore di traffico e simulatore di rete, ma è il primo sistema open-source che prevede l'invio di messaggi dal simulatore di rete al simulatore di traffico per influenzare la mobilità dei veicoli in base

ai messaggi utilizzati. Nella figura 2.2 è mostrata l'architettura di TraNS: l'applicazione VANET è implementata nel simulatore di rete, e comunica con un modello (*driver behaviour model*) a cui passa le proprie informazioni. Questo, in base alle informazioni ricevute, invia a SUMO dei comandi predefiniti (*atomic mobility commands*) per influenzare la mobilità, come ad esempio *stop*, *change lane*, *change speed*. Purtroppo, TraNS ha presto smesso di essere attivamente supportato perdendo gradualmente interesse [8].

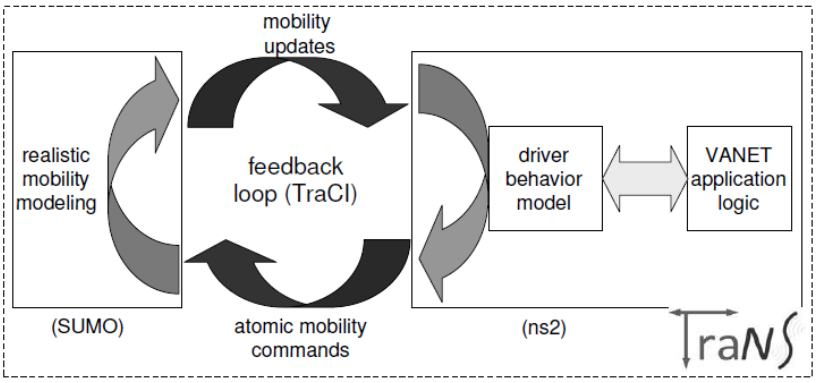


Figura 2.2: Architettura di TraNS in modalità application-centric, con scambio di dati bidirezionale tra i simulatori (immagine presa da [5]).

Un altro sistema integrato è Veins (Vehicles In Network Simulation [7]), che integra SUMO con OMNeT++, quest'ultimo scelto per il suo INET framework. Quando SUMO ritorna le informazioni sul movimento dei veicoli, OMNeT++ le utilizza per aggiornare la posizione dei moduli dei veicoli.
L'interazione tra i due sistemi è mostrata in figura 2.3. Il simulatore di rete fa avanzare SUMO a intervalli fissi di tempo simulato (time step), inviandogli il tempo della simulazione nel messaggio di sincronizzazione. Nel corso del suo time step, OMNeT++ invia a SUMO vari comandi attraverso una connessione TCP, mentre SUMO li memorizza in un buffer attendendo il proprio tempo di esecuzione per eseguirli. Al termine del propio time step, SUMO ritorna a OMNeT++ le informazioni di mobilità.
La simulazione di SUMO può essere influenzata da OMNeT++ in vari modi: ad esempio, i veicoli possono essere fermati per simulare malfunzionamen-

ti o ingorghi, o essere dirottati in tratti stradali differenti. Inoltre, nuovi veicoli possono essere creati o rimossi, per esempio una volta raggiunta la destinazione.

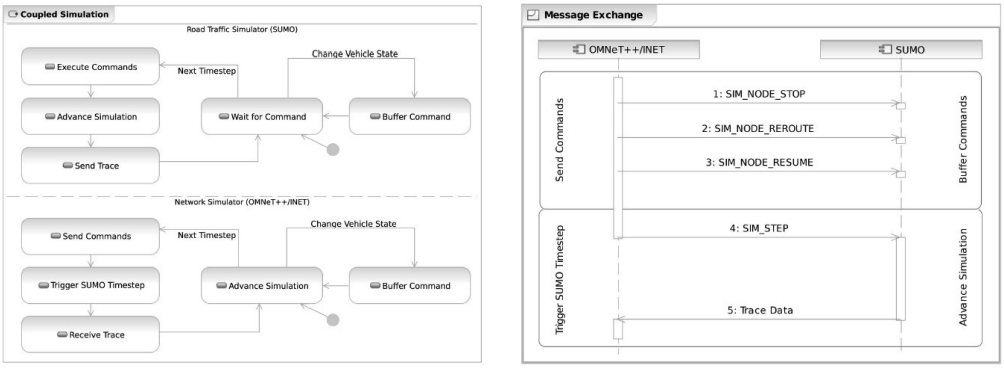


Figura 2.3: A sinistra, il flusso di esecuzione dei singoli simulatori. A destra, esempio dove i due simulatori si alternano nell'esecuzione di un time step, con OMNeT++ che invia i comandi da eseguire prima di eseguire la sincronizzazione. (immagine presa da [7]).

Mentre TraNS e Veins fanno comunicare direttamente i due simulatori, iTETRIS [8] utilizza un'architettura flessibile a 3 blocchi (figura 2.4) introducendo come punto intermedio di comunicazioni iCS (iTETRIS Control System). Le applicazioni per VANET che devono essere testate utilizzano le interfacce fornite da iCS, rendendo gli sviluppatori di tali applicazioni indipendenti dai simulatori e permettendo di utilizzare un qualsiasi linguaggio di programmazione (purchè supporti la comunicazione tramite socket).
iTETRIS è un progetto supportato dall'Unione Europea, ed è conforme all'architettura di comunicazione specificata da ETSI (European Telecommunications Standards Institute) per i Sistemi di Trasporto Intelligenti (ITS). iTetris utilizza SUMO e NS-3 [9], e può essere visto come architettura logicamente successiva a TraNS. Uno dei requisiti di iTETRIS è quello di progettare strategie che portino benefici ambientali, come ad esempio ridurre le emissioni inquinanti dei veicoli e il consumo di carburante, ma ci sono anche altri scopi che richiedono visione o controllo di alcuni aspetti della simulazione,

ad esempio lo stato dei semafori; SUMO è stato quindi esteso per ottenere tali obiettivi creando interfacce per le funzioni desiderate utilizzabili dall'applicazione tramite iCS.
Un altro requisito di iTETRIS è quello di simulare grandi quantità di veicoli (decine di migliaia) e i simulatori SUMO e NS-3 sono stati scelti apposta per la loro capacità di gestire tale quantità di elementi. Tuttavia, a causa delle simulazioni che si vuole condurre con iTETRIS (scenari di intere città e veicoli che usano più tecnologie di comunicazioni nello stesso momento), si è effettuato varie ottimizzazioni a NS-3 per migliorare le performance, principalmente andando a semplificare la complessità del livello fisico.

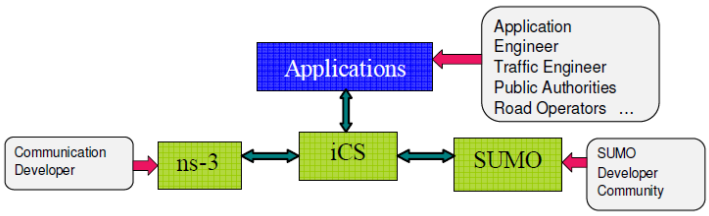


Figura 2.4: Architettura di iTETRIS: interazione tra i componenti del sistema e rapporto tra componenti e utenti/sviluppatori (immagine presa da [8]).

I casi di integrazione qui riportati uniscono simulatori di rete con simulatori di traffico, con lo scopo di effettuare una simulazione più accurata delle applicazioni per VANET che vanno a influenzare la mobilità dei veicoli; questi strumenti sono molto utili, in quanto i test reali sono costosi da effettuare e non possono essere fatti per un numero di veicoli elevato, come invece è necessario per ottenere risultati significativi. Tuttavia, mentre i simulatori di traffico vengono usati per dare al simulatore di rete informazioni di mobilità per i veicoli della simulazione attraverso l'uso di un modello realistico, quello che si cerca di fare con il lavoro di questa tesi è fornire non gli spostamenti ma i dati dei sensori, simulando accuratamente i movimenti e i sensori dei droni in un ambiente realistico e visualizzando la simulazione mediante grafica 3D.

2.1.2 Integrazione tra simulatori di robot e di rete

Un caso di integrazione di vari strumenti è descritto in [10](HNMSim, 2009-2012). Il sistema creato è stato pensato come strumento di studio e di educazione relativamente ai sistemi multi-agente, (networked multi-agent systems) e nel documento si fa riferimento anche agli UAVs. Per simulare ambiente, movimento dei robot e sensori utilizza USARSim (Unified system for automation and robot simulation), che fa uso dell'Unreal Engine; l'ambiente di simulazione può essere creato attraverso l'Unreal Editor. Per il controllo dei robot vengono usati LabView e Matlab, mentre per la gestione della rete di comunicazione si fa uso di OMNeT++.
Essendo il sistema diviso in rete, controllo e ambiente, c'è molta comunicazione tra i componenti. Per quanto riguarda la simulazione degli UAVs, è generalmente difficile, anche se possibile, implementare le dinamiche di un UAV in Unreal, e per questo si è affidato parte del compito a Matlab (figura 2.5). Un problema del sistema è che Matlab non è libero da usare, anche se potrebbe essere disponibile tramite il contesto universitario.

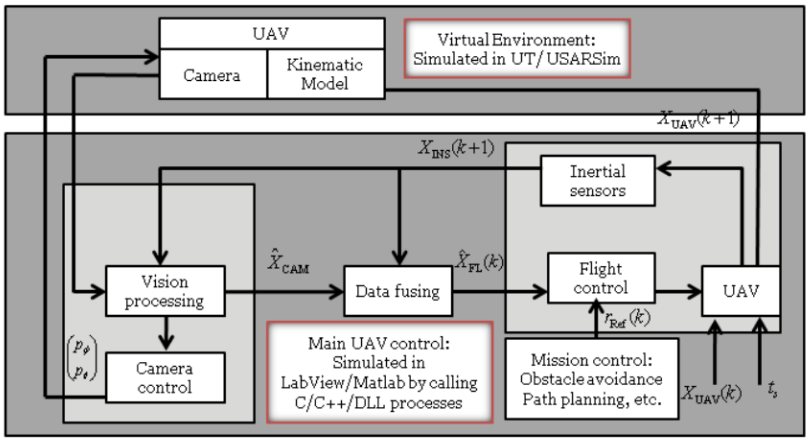


Figura 2.5: La struttura di controllo di un UAV con sensore visivo in HNMSim (immagine presa da [10]).

Mentre HNMSim è composto da vari componenti, altri sistemi presentano una struttura più semplice. Ad esempio, un progetto riguardo all'unione

di due simulatori è [11](RoboNetSim, 2013), dove si uniscono ARGoS, un simulatore pensato per i gruppi di robot, e NS-2/NS-3, simulatori di rete. ARGoS comunica al simulatore di rete il pacchetto da trasmettere e riceve da lui i pacchetti arrivati da altri nodi. ARGoS si occupa di muovere i robot, poi passa le nuove posizioni al simulatore di rete, che aggiorna le posizioni dei nodi di rete. Ogni nodo di rete presenta un'applicazione che dialoga con il corrispondente robot di ARGoS tramite socket, e la comunicazione tra i due non è limitata ai pacchetti, ma può prevedere altri dati di interesse come ad esempio gli ostacoli presenti nell'ambiente. Tuttavia, ARGoS non fornisce modelli adatti per il volo di UAVs, e per come RoboNetSim è composto non è adatto alla loro simulazione [12].

Un sistema integrato pensato apposta per effettuare simulazioni di UAVs è descritto in [12](CUSCUS, 2018). Per questo scopo sono stati scelti FL-AIR (Framework Libre Air, [13]) per la simulazione del drone e NS-3 per la comunicazione con gli altri droni. Inoltre, il sistema fa uso di OpenStreetMap per generare l'ambiente di simulazione, che è utilizzato da un modulo creato in NS-3 per determinare l'attenuazione dei segnali causata dagli ostacoli.
Per integrare i due simulatori si è fatto uso della modalità *real-time* di NS-3, che rende possibile far interagire il simulatore di rete con le interfacce di rete della macchina su cui è eseguito. NS-3 crea delle interfacce di rete virtuali attraverso le quali riceverà i dati provenienti da FL-AIR. FL-AIR è composto da un insieme di programmi indipendenti che comunicano tra di loro mediante socket. Ogni drone ha il suo processo, e in CUSCUS ognuno di questi processi viene eseguito all'interno di un Linux Container, che viene usato per intercettare i pacchetti di rete del processo ed inviarli tramite un bridge di rete all'interfaccia virtuale creata da NS-3.
CUSCUS permette di simulare in maniera accurata il comportamento degli UAVs. Inoltre, il passaggio da simulatore a test sul campo può avvenire velocemente in quanto lo stesso codice dell'applicazione del drone usato nel simulatore può essere usato su un drone reale. Un problema di usare FL-AIR, però, è che il mantenere un processo per ogni drone della simulazione

rende il sistema poco scalabile.

Un altro caso di integrazione più recente è presentato in [14](COPADRIVe, 2019): in questo lavoro si sono uniti gli stessi ambienti utilizzati in questa tesi, cioè OMNeT++ e Gazebo, ma in questo caso per l'organizzazione di un gruppo di automobili per lo studio di una Vehicle Ad-hoc Network (VANET). Ciò che si è realizzato nella tesi è simile al sistema presentato in questo articolo: la comunicazione tra OMNeT++ e Gazebo avviene attraverso i topic e si utilizza un modulo di OMNeT++ per la sincronizzazione. L'idea del modulo sincronizzatore é stata infatti presa da questo articolo.

In generale, il sistema creato si differenzia dai sistemi citati per un aspetto: c'è un unico punto di contatto tra i due simulatori. In tutti gli altri sistemi la comunicazione tra simulatore di rete e di controllo del drone avvengono in maniera individuale per ogni drone, mentre nel sistema creato sono i due simulatori che si scambiano tutti i dati della simulazione a loro necessari (figura 2.6). Questo è dovuto al fatto che, in maniera simile a COPADRIVe, il controllo del singolo robot viene gestito all'interno del simulatore di rete (OMNeT++ in entrambi i casi).

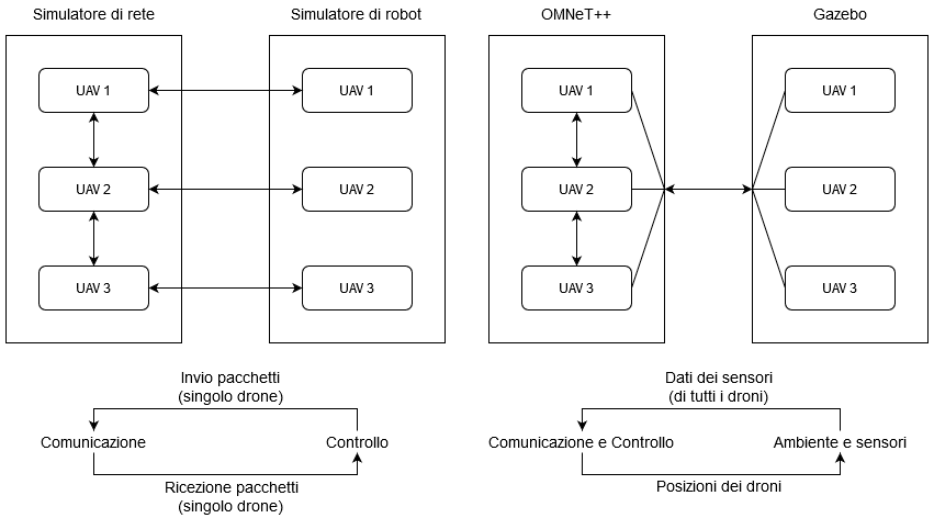


Figura 2.6: Differenza nel rapporto tra simulatore di rete e simulatore del drone/robot. La parte di destra rappresenta il sistema creato in questa tesi.

Un'altra differenza tra il sistema creato e COPADRIVe è tuttavia presente nella gestione della sincronizzazione tra i due sistemi: COPADRIVe, infatti, utilizza un modulo sincronizzatore in OMNeT++ che si registra a un topic di Gazebo che scandisce il tempo (e quindi è Gazebo che conduce la simulazione) mentre il modulo di sincronizzazione del sistema creato utilizza il tempo di OMNeT++, e Gazebo può continuare la sua simulazione solo quando OMNeT++ ha finito di gestire i suoi eventi (OMNeT++ conduce la simulazione).
Mentre la presenza di un singolo punto di contatto tra i due simulatori e l'accorpamento della parte di logica e comunicazione in un unico simulatore semplificano il sistema, il fatto di alternare la simulazione tra OMNeT++ e Gazebo può portare a problemi di performance, in quanto uno dei due sistemi è sempre in attesa; tuttavia, attendere che l'altro simulatore finisca la sua parte di simulazione dovrebbe portare a una simulazione più fedele al caso reale.

2.2 Swarm intelligence per il movimento dei droni

Si può pensare a diversi modi di organizzare un gruppo di nodi allo scopo di ottenere da loro un certo comportamento o funzionalità. Un modo di organizzarli e dirigerli è quello di mettere un leader al comando, che prenda tutte le decisioni e impartisca ordini, creando così un sistema centralizzato. In questo modo, vengono passate al leader le informazioni della rete, cioè le informazioni dei singoli nodi, e questo si ritrova con una visione completa del sistema (o comunque con tutte le informazioni di interesse del problema). Avendo quindi tutti i dati in un unico punto è possibile prendere decisioni migliori. Un problema di questo approccio è però la tolleranza ai guasti, in quanto il fallimento del leader blocca il sistema, a meno che questo non possa essere sostituito. Un altro problema riguarda la scalabilità del sistema: con un numero di nodi crescente una soluzione di questo tipo diventa inefficiente.

È possibile risolvere i problemi di tolleranza ai guasti e scalabilità rendendo il sistema distribuito, evitando di decidere ogni azione in un singolo punto e decidendo come agire in base a informazioni locali. Un fenomeno di questo tipo si può osservare in natura nei gruppi di animali, dove i singoli individui prendono decisioni autonomamente mostrando però un comportamento collettivo che porta al raggiungimento di un certo scopo.
A questo punto, singoli nodi dalle limitate capacità possono mostrare comportamenti complessi attraverso semplici regole basate su informazioni locali. Dato che ogni nodo prende decisioni autonomamente, il fallimento di uno di questi non compromette necessariamente la funzionalità del sistema e la scalabilità aumenta. Un problema diventa però il riuscire a trovare le regole di comportamento dei nodi che portino ad avere un certo comportamento globale, e risulta necessario effettuare test per trovare i valori dei parametri del sistema che portino ai risultati desiderati. Inoltre, a causa del controllo distribuito, con determinati valori dei parametri o scenari potrebbero avvenire comportamenti inaspettati.

In questa tesi si è deciso di studiare il comportamento di droni organizzati mediante l'approccio distribuito. Ogni drone basa il suo movimento su alcune informazioni locali: la vicinanza con gli altri droni e l'ambiente circostante.

La forma di mobilità adottata (descritta nel capitolo Capitolo 4) fa uso di molle virtuali, presenti tra coppie di droni, che attraggono o respingono i droni a seconda della potenza del segnale ricevuto, con lo scopo di garantire una certa qualità del servizio. Vengono quindi presentati alcuni studi riguardo le molle virtuali, tra cui quello a cui si è fatto riferimento per questa tesi, e in seguito altri due sistemi per l'organizzazione di uno sciame di nodi, sempre ispirati alla fisica.

2.2.1 Molle virtuali

La tecnica delle molle virtuali facente uso del requisito di qualità del servizio è stata presa da [15](2015), dove un gruppo di droni terrestri a mobilità autonoma ha lo scopo di formare una rete di comunicazione provvisoria

di emergenza in uno scenario dove l'infrastruttura di rete esistente è stata compromessa a seguito di un qualche disastro. Un obiettivo del gruppo di droni è quello di esplorare lo spazio alla ricerca di dispositivi da connettere, mantenendo il gruppo interconnesso. Vengono utilizzati tre tipi di molle: mesh-to-mesh (per il collegamento dei droni), mesh-to-EU (End User, i dispositivi che hanno bisogno di collegarsi alla rete) e mesh-to-Frontier (molle che spingono dei droni *scout* a esplorare l'area).

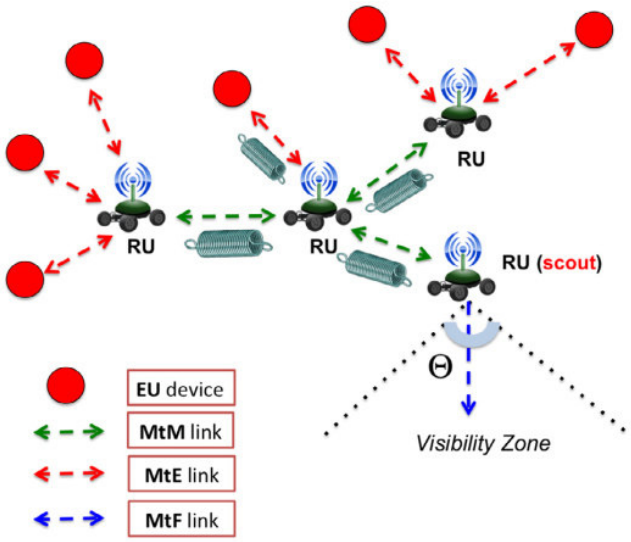


Figura 2.7: Repairing Units (RU) e i vari tipi di molle virtuali che ne determinano il movimento (immagine presa da [15]).

Il precedente studio va ad estendere l'idea di virtual spring mesh (VSM), come anche il lavoro presentato in [16](2011). In questo articolo si porta il gruppo di nodi a esplorare un ambiente in modo da coprire l'area disponibile ed evitare ostacoli fissi e mobili. Anche qui ci sono dei nodi che conducono l'esplorazione: i nodi esterni alla mesh vengono spinti a espandere la mesh attraverso due forze, la forza di espansione (che li allontana dai nodi vicini) e la forza di esplorazione (che li porta a muoversi verso gli ostacoli più lontani). Nel calcolo della forza di esplorazione viene usata una molla per ostacolo e a ogni ostacolo viene assegnato un peso a seconda della distanza (peso maggiore agli ostacoli più lontani). L'algoritmo eseguito dai nodi per determinare

il loro movimento porta inoltre ad evitare le collisioni in quanto evita di avvicinarsi troppo a un ostacolo e spinge ad allontanarsi da esso se la distanza tra i due è minore rispetto a un valore specificato.

Il metodo delle molle virtuali è stato precedentemente studiato in [17](2005), dove si indaga i diversi metodi per la creazione delle molle; per esempio, un drone potrebbe creare delle molle con ogni altro drone che riesce a percepire attorno a sè, oppure con solo un sottoinsieme di questi droni, scelti per vicinanza o altri criteri. Tra questi metodi si indaga in particolare quello che viene ritenuto più adatto allo scopo, il test dell'angolo acuto, che è stato utilizzato anche in [16] e in questo progetto (vedi la sottosezione 4.2.1).

Sciame guidato da un leader

Mentre in [15] e [16] lo spostamento del gruppo di nodi nell'ambiente è guidato da nodi ai margini del gruppo, nel sistema creato per la tesi la migrazione dello sciame da un luogo a un altro avviene utilizzando un nodo leader che ne guida il movimento; si è usato quindi un metodo non tollerante ai guasti, in quanto il fallimento del leader fermerebbe la migrazione, se nessun altro nodo prendesse il suo posto.
Il caso di un gruppo di nodi organizzato tramite molle virtuali e di un nodo leader che ne decide e guida lo spostamento è già stato considerato in alcuni studi. In [18](2016) si è esaminato, tramite esperimento reale con tre robot Turtlebot, il caso dove alcuni robot si organizzano attorno a un leader attraverso due tipi di molle virtuali: tra Turtlebot e leader, per mantenere una certa distanza dal leader, e tra coppie di Turtlebots, per mantenere una certa di distanza dagli altri followers. Il sistema utilizza le coordinate dei robot per determinare la forza generata delle molle; mentre per il caso reale si è pensato all'uso del GPS, nell'esperimento questo è stato simulato utilizzando un drone posto sopra la zona del test. Un computer riceve le informazioni di posizione dal drone e introduce la posizione di un leader virtuale; calcola quindi la forza delle molle e comunica ai Turtlebots i comandi per muoversi secondo tale logica. Gli esperimenti hanno mostrato come i Turtlebots mantengano

la formazione per traiettorie semplici, ma come per traiettorie più complesse fossero presenti dei problemi, in parte legati alla modalità dell'esperimento.

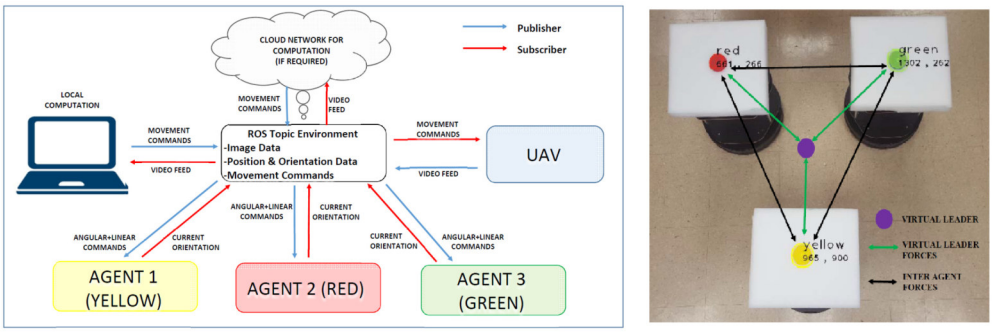


Figura 2.8: Immagini prese da [18]. A sinistra l'architettura del sistema, a destra i Turtlebots e le forze che agiscono tra loro e il leader.

Un altro caso in cui i robot si organizzano attorno al leader è visto in [19](2019). In questo articolo lo scopo è quello di utilizzare una formazione di holonomic robots per operazioni di trasporto, che ha il vantaggio di riuscire a trasportare quello che un singolo robot non potrebbe e fornisce affidabilità, flessibilità e tolleranza ai guasti. Come nel precedente articolo sono presenti molle tra coppie di followers e tra follower e leader (vedi figura 2.9). In questo esperimento, però, i robot followers utilizzati sono quattro, che mantengono molle follower (inter agent force) solo con i più vicini tra di loro (ognuno mantiene solo due molle follower, ignorando il robot più lontano). Inoltre, viene usato un altro tipo di molla tra follower e leader (rotational virtual leader force, F^{rot} nell'immagine) il cui scopo è quello di far mantenere ai robot lo stesso orientamento del leader.
Il metodo è stato implementato usando ROS [28] e valutato utilizzando Gazebo. Negli esperimenti si sono effettuati due tipi di test: la composizione della formazione partendo da posizioni dei robot casuali e il mantenimento di tale formazione durante i diversi tipi di spostamento. I risultati mostrano come la formazione venga mantenuta con errori di posizionamento trascurabili.

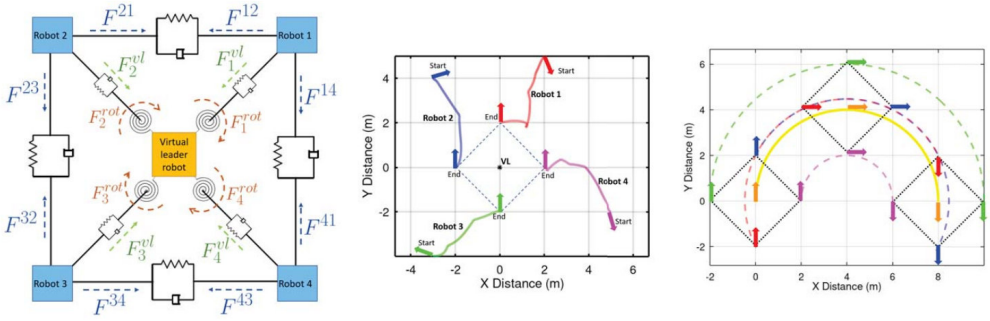


Figura 2.9: Immagini prese da [19]. Da sinistra, le molle usate dai followers, la creazione della formazione per il trasporto, il movimento coordinato in caso di curva.

Mentre negli articoli sopra citati i robot gestiscono l'evitamento della collisione solo con gli altri robot (l'ambiente non è considerato) e si dispongono in formazione attorno al leader, in [20](2008) si descrive una situazione più simile al caso di studio di questa tesi, ovvero un robot leader che guida un gruppo attraverso un percorso che può mostrare la presenza di ostacoli che è necessario evitare. Anche qui i robot sono disposti tramite una formazione, che però viene alterata in base alle condizioni dell'ambiente per poi essere ricomposta dove possibile.
I followers seguono il leader mediante delle molle virtuali che possono essere presenti tra coppie di loro e tra loro e il leader (vedi figura 2.10). Inoltre dispongono di molle per mantenere un certo angolo nella formazione rispetto ad altri robot. Anche il leader è legato ai follower tramite molle, ma è influenzato anche da una forza ricavata da un'operazione di ricerca del percorso (G_L nell'immagine). Il leader ha infatti il compito di condurre il gruppo di robot verso una certa destinazione obiettivo, ed è suo incarico trovare il percorso migliore.

Per quanto riguarda gli ostacoli, i robot utilizzano un sensore visivo per rilevare quelli presenti vicino a loro. Considerano gli ostacoli come linee che in parte li respingono lungo la perpendicolare (forza respingente, F_r) ma in parte li spingono ad aggirare l'ostacolo andando verso la posizione del lea-

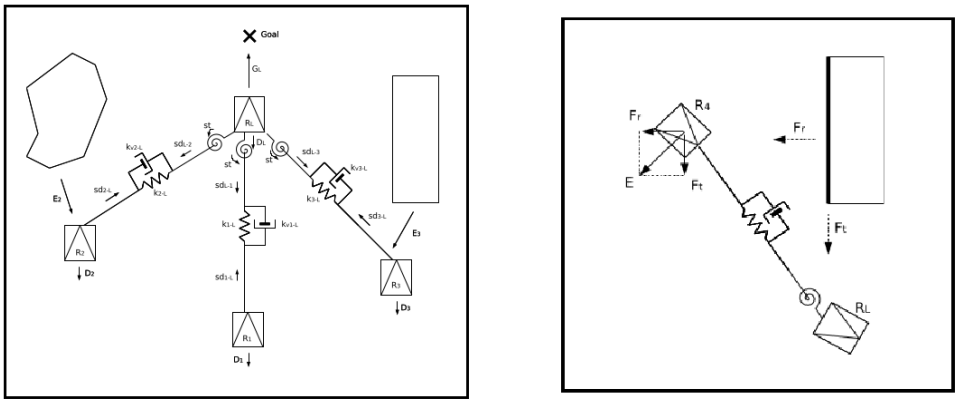


Figura 2.10: Immagini prese da [20]. A sinistra le forze che agiscono sui robot, a destra le forze generate dagli ostacoli.

der (forza tangente, F_t). Bisogna stare attenti a quest'ultima componente in quanto in certi scenari ha l'effetto di accellerare il movimento dei follower, e in questi casi nello studio effettuato tale forza viene ignorata.
Test del sistema sono stati fatti sia tramite simulazione che tramite dispositivi reali. Al computer sono stati esaminati degli scenari con curve, restringimento dello spazio disponibile e ostacoli, superati senza presenza di collisioni. Si è poi mandato tre robot Pioneer 3AT lungo un corridoio con un restringimento nel mezzo che permette solo a un robot alla volta di passare, e si è visto come anche nella realtà i followers rallentano e passano prima uno e poi l'altro mentre il leader rallenta per aspettarli.

Mentre negli articoli sopra citati i followers si organizzano attorno al leader o lo seguono cercando di mantenere una certa formazione, in questa tesi i followers si organizzano utilizzando il test dell'angolo acuto [17], senza la ricerca di una determinata formazione da mantenere o da ricomporre alla fine dello spostamento. Per quanto riguarda l'evitamento degli ostacoli, l'idea della forza tangente per aggirare l'ostacolo è interessante ma si è scelto di verificare un sistema più semplice, ma che ignora la direzione di viaggio, basato su una approssimativa posizione dell'ostacolo (metodo descritto nella

sezione sezione 4.3).

2.2.2 Altri sistemi ispirati alla fisica

Oltre alle molle, altri sistemi di fisica virtuale sono stati studiati per l'organizzazione dei robot, ispirati alle cariche elettriche [21] o alla forza gravitazionale [22, 23, 24].

In [21] il metodo studiato ha l'obiettivo di spargere i nodi, componenti di una rete di sensori, nell'ambiente in cui vengono rilasciati. I nodi hanno un sensore per percepire l'ambiente circostante, e il loro movimento dipende dagli ostacoli e dagli altri nodi presenti nelle loro vicinanze.
I nodi sono trattati come particelle virtuali soggette a forze virtuali, che li respingono dagli ostacoli e dagli altri nodi. Le forza che un nodo o un ostacolo esercita su un dato nodo è data dalla distanza tra i due, non dalla posizione assoluta nell'ambiente, che può quindi anche essere sconosciuto.

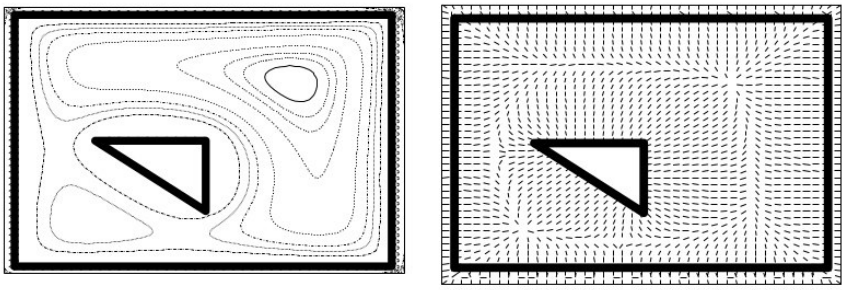


Figura 2.11: I campi elettrici generati da un ambiente semplice e la direzione della forza da essi generata (immagine presa da [21]).

Una proprietà di questo approccio è quello di dimostrare il raggiungimento di un equilibrio (nodi che smettono di muoversi) pensando al sistema come composto da energia potenziale e cinetica, data dalla forza virtuale che l'ambiente esercita sui nodi e dal loro movimento. Nella formula che determina il movimento dei nodi è presente un elemento di frizione (viscous friction term) che ha l'effetto di rimuovere energia dal sistema, che quindi potrà solo dimi-

nuire nel tempo fino ad annullarsi. Questo nel caso che l'ambiente non cambi nel tempo, perchè altrimenti porterebbe nuovamente energia nel sistema.

In [22] si esamina invece dei metodi di formazione di topologie per sciami di micro-air vehicles (MAVs) che, data la natura dei dispositivi (con semplici sensori e CPU), utilizzi semplici regole per determinare il posizionamento dei nodi. I nodi utilizzano i sensori per percepire i loro vicini, e vengono attratti o respinti (come per le molle) a seconda della distanza tra di loro rispetto a uno specificato valore R. La formula che determina la forza di attrazione/repulsione è data dalla formula di gravitazione universale, con G come parametro del sistema.
Attraverso questo metodo si ottiene una formazione a esagono, perchè il centro dell'esagono e i suoi vertici sono tutti a distanza R tra di loro. Vengono però esaminati altri metodi e comportamenti: la formazione di cluster di nodi e il loro rapporto con il parametro G, la formazione di una topologia a base quadrata assegnando ai nodi un valore di "spin", il cambiamento del valore di "spin" nei nodi dei cluster, la disposizione dei nodi in topologia attraverso una operazione di ordinamento nello spazio.

Capitolo 3

Architettura

In questo capitolo vengono presentati i due simulatori (OMNeT++ e Gazebo) e il sistema creato che prevede la loro integrazione. Come si è scritto nell'introduzione, obiettivo della tesi è la creazione di un sistema integrato dove i due simulatori partecipano all'esecuzione di una stessa simulazione condivisa, che permetta lo studio della forma di mobilità di uno sciame di droni. In particolare, in OMNeT++ viene definita ed eseguita la logica di comportamento del drone (gestione del movimento e comunicazione) mentre Gazebo si occupa della riproduzione 3D della simulazione e della gestione dell'ambiente (gli ostacoli e i sensori).

3.1 OMNeT++

OMNeT++ (Objective Modular Network Testbed in C++) è uno strumento tramite il quale è possibile condurre simulazioni e raccogliere i dati da esse generati. È diventato noto come simulatore di reti, ma in realtà non lo è: è stato sì creato con in mente l'utilizzo per la simulazione di reti di comunicazioni e altri sistemi distribuiti ma, invece di costruire un simulatore specializzato, OMNeT++ è stato progettato per essere il più generale possibile. È utilizzabile gratuitamente per simulazioni non commerciali.

OMNeT++ presenta un ambiente di sviluppo integrato (IDE) basato su Eclipse, dove definire il sistema da simulare e i suoi componenti. Gli elementi della simulazione sono i *moduli*, di cui è possibile definire il comportamento utilizzando il linguaggio C++. I moduli possono essere semplici o composti (possono contenere altri moduli) e possono quindi rappresentare un elemento di alto livello, come un nodo di rete, o i "mattoncini" di cui questo è composto.

La comunicazione tra i vari componenti avviene principalmente tramite l'invio di messaggi, che viaggiano lungo i loro collegamenti. I collegamenti tra i vari moduli vengono definiti in un file NED (NEtwork Description), dove si inseriscono gli elementi della simulazione e i loro parametri. I file .ned sono anche usati per definire la struttura e i parametri di un singolo modulo oltre al codice C++.

I file .ned vengono scritti in un loro apposito linguaggio, ma attraverso l'IDE è possibile anche vedere una rappresentazione grafica di ciò che è stato definito. La stessa grafica viene mostrata al momento della simulazione, dove permette di capire alcuni suoi avvenimenti grazie all'aggiunta di alcuni effetti grafici, per esempio:

- messaggi, rappresentati da pallini rossi, che passano da modulo a modulo
- invio di pacchetti, i cui dati formano linee rosse che viaggiano verso destinazione
- spostamento di moduli mobili (es. droni o smartphone)

I controlli di simulazione permettono di controllarne l'esecuzione: si può eseguire la simulazione a velocità differenti, fermarla, o eseguire solo il prossimo evento della coda.

OMNeT++, infatti, organizza la simulazione in eventi, che hanno luogo in un preciso istante di tempo; quando un modulo ne influenza un altro, ad esempio inviandogli un messaggio, comunica ad OMNeT++ quando il

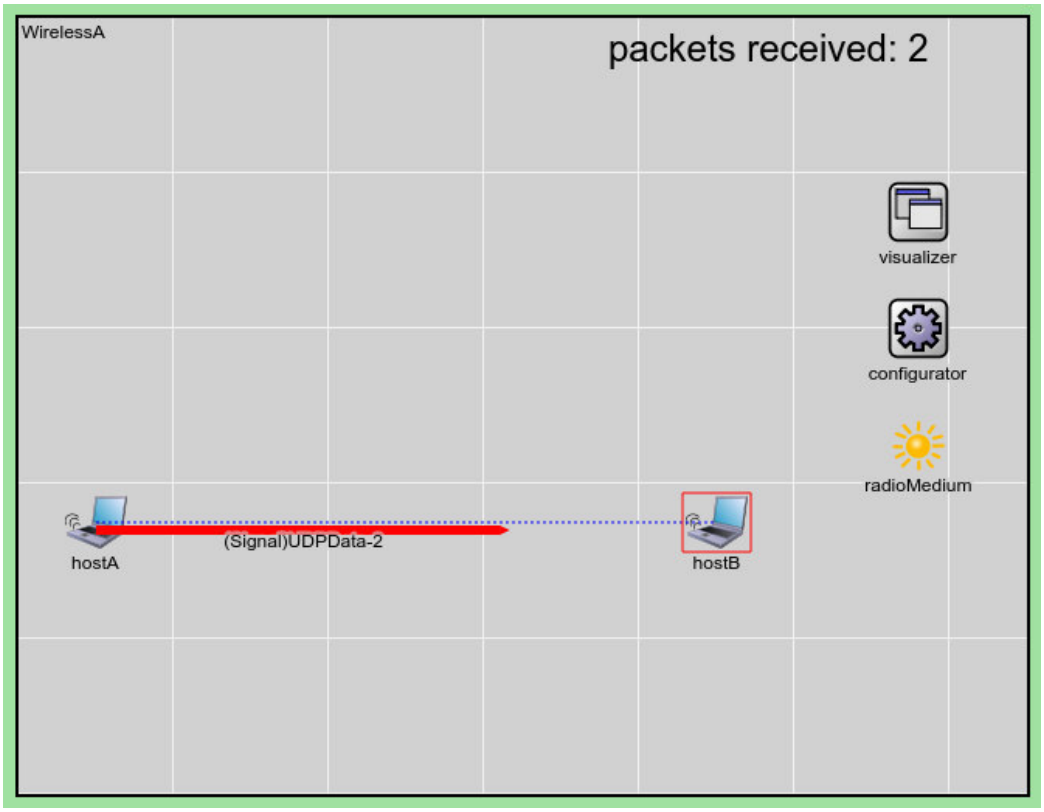


Figura 3.1: Simulazione in OMNeT++ dove un host invia all'altro un pacchetto. Il flusso di dati è mostrato tramite una linea rossa.

messaggio deve essere processato da quel modulo. La stessa cosa accade quando un modulo programma una sua prossima azione futura, per esempio un nuovo tentativo di accedere a un canale di comunicazione per l'invio di un messaggio.

OMNeT++ prende questi eventi e li accumula in una coda, chiamata FutureEventSet (FES), dove li ordina in base al tempo di avvenimento. Quindi, a ogni passo di simulazione, OMNeT++ prende il primo evento della coda (il prossimo che deve essere eseguito) e lo consegna al modulo a cui era destinato, attendendo che questo finisca il suo lavoro in risposta a tale evento.

Ogni modulo contiene una funzione che viene eseguita in risposta alla ricezione di un messaggio (*handleMessage(..)*). Non tutta l'interazione però è governata dai messaggi: i moduli possono interagire tra di loro chiamando

direttamente le funzioni dell'altro.

OMNeT++ di per sè è un ambiente di simulazione, e non comprende modelli per architetture e protocolli di rete, ma nel corso degli anni sono state fatte molte simulazioni e molti modelli sono stati creati, alcuni liberamente disponibili.

L'INET framework [25], utilizzato in questo progetto di tesi, è una libreria open-source che contiene protocolli, agenti e altri modelli utilizzabili da ricercatori e studenti che vogliono lavorare con reti di comunicazioni. Il framework è mantenuto dal team di OMNeT++ interagendo però anche con i membri della sua community. L'INET framework si può considerare la libreria standard di protocolli di OMNeT++ e costituisce una base per diversi altri framework di simulazione.

3.2 Gazebo

Gazebo è un simulatore 3D open-source in grado di simulare accuratamente ed efficientemente gruppi di robot. Fornisce il supporto a 4 diversi physics engine (ODE, Bullet, Simbody e DART), un'ampia selezione di sensori, una libreria di modelli di robot e elementi ambientali e la possibilità di estendere alcune sue funzionalità tramite la creazione di plugin.

La simulazione agisce su dei modelli, che vengono definiti in file con formato SDF, un formato XML nato per l'utilizzo in Gazebo. All'interno di questi file si definiscono le parti del modello, i sensori che utilizza e varie proprietà.

I modelli possono essere caricati direttamente nel simulatore attraverso interfaccia grafica o indirettamente attraverso il caricamento di file .world, cioè file definiti in SDF che indicano i modelli da caricare e la loro posizione nello spazio.

Gazebo divide il simulatore in due programmi, server e client:

- gzserver, dove avviene la simulazione

- gzclient, interfaccia grafica per visualizzare la simulazione e interagire con il server

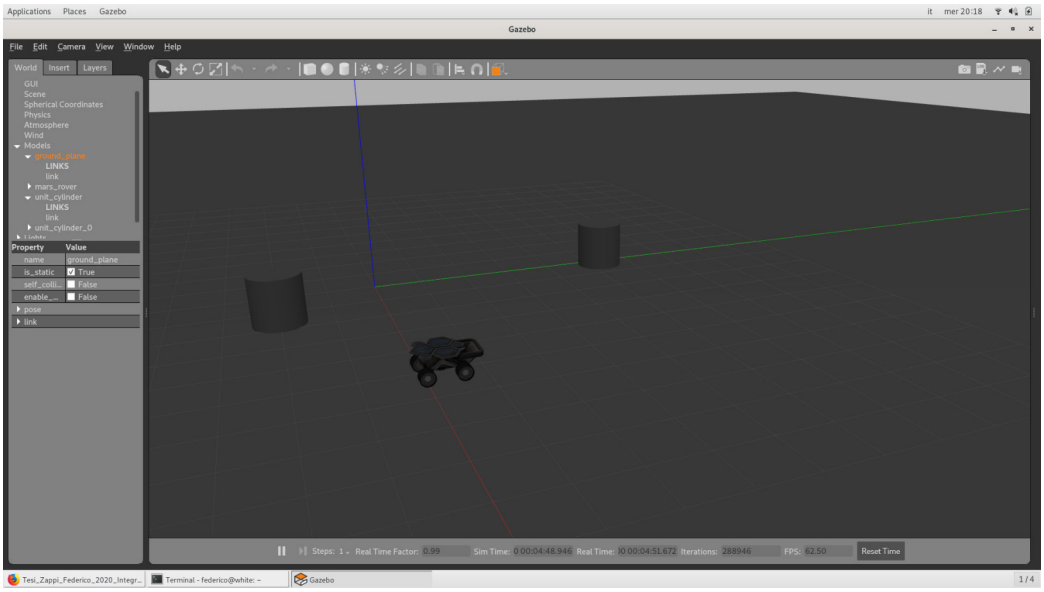


Figura 3.2: L'interfaccia grafica di Gazebo.

Client e server comunicano utilizzando la libreria di comunicazione di Gazebo, che fa uso di Google ProtoBuf (Protocol Buffers [26]) per la serializzazione dei messaggi e boost::ASIO [27] per il meccanismo di trasporto. La forma di comunicazione è publish/subscribe: si fa subscribe ad un topic per ricevere messaggi quando qualcuno vi farà publish.

Le librerie di rendering, fisica e sensori supportano l'uso di plugin, che permettono di accedere alle librerie e modificare o estendere la loro funzione senza utilizzare il meccanismo di comunicazione. Per indicare a gazebo di utilizzare un certo plugin si può fare riferimento a lui nel comando di avvio di gazebo o all'interno di un file .wordl o di un file di un modello.

Lo sviluppo di Gazebo è cominciato a fine 2002 in ambito universitario. Il nome è stato scelto pensando agli ambienti esterni come ambiente di simulazione principale, ma di fatto è maggiormente utilizzato per simulazioni in ambienti interni. Dal 2009 Gazebo permette l'integrazione con ROS (Robot Operating System [28]) e da allora è uno degli strumenti chiave della

ROS community. Al momento, il suo sviluppo è guidato dalla Open Source Robotics Foundation (OSRF).

3.3 Unione dei due sistemi

Come si è detto precedentemente nell'introduzione, lo scopo primario della tesi era quello di far collaborare OMNeT++ e Gazebo nell'esecuzione di una simulazione, avendo tramite Gazebo la visualizzazione 3D dello scenario e la generazione dei dati dei sensori, e tramite OMNeT++ il resto del lavoro, tra cui la gestione della mobilità dei droni con il calcolo delle loro posizioni nell'ambiente.

3.3.1 L'alternanza nella simulazione

In particolare, il seguente schema descrive il rapporto tra i due simulatori.

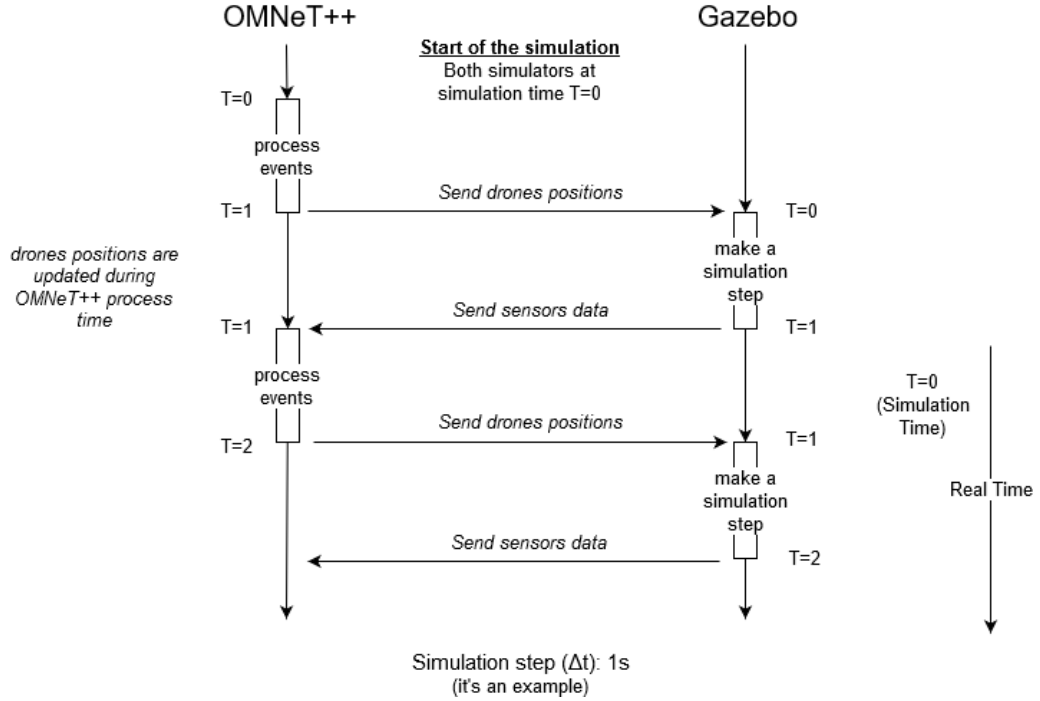


Figura 3.3: Schema di interazione tra OMNeT++ e Gazebo con passo di simulazione Δt di 1 secondo

Dato un certo intervallo di sincronizzazione Δt e il tempo di simulazione T:

- (tempo T) OMNeT++ processa gli eventi presenti nella coda degli eventi fino ad arrivare al tempo di simulazione T+Δt

  (tempo T) Gazebo attende

- (tempo T+Δt) OMNeT++ invia le posizioni aggiornate dei droni a Gazebo

  (tempo T) Gazebo riceve il messaggio di OMNeT++ ed esegue un passo Δt di simulazione

- (tempo T+Δt) Gazebo invia gli eventuali dati dei sensori a OMNeT++

  (tempo T+Δt) OMNeT++ riceve il messaggio di Gazebo e può ricominciare la simulazione

Quello che accade è che i due simulatori si alternano nell'esecuzione una volta eseguito un certo passo di simulazione. Inoltre, come si può vedere, OMNeT++ è il primo ad eseguire il passo di simulazione successivo, che porta avanti la simulazione.

Nel momento in cui il simulatore "attivo" deve cedere il passo a quello in attesa, oltre ad avvertire l'altro simulatore che è il suo turno di agire vengono passati anche i dati ad esso necessari; **il passaggio dei dati all'altro simulatore avviene attraverso un singolo messaggio**, cioè i dati dei vari droni sono tutti contenuti in un unico messaggio. Questo unico punto di contatto è stato richiesto in quanto rende il sistema più semplice e più facilmente modificabile in eventuali sviluppi futuri.

3.3.2 Gli elementi principali

L'architettura del sistema integrato è mostrata nella figura 3.4. Nell'immagine si possono vedere i due simulatori (OMNeT++ e Gazebo), alcuni loro componenti e l'interazione tra questi.

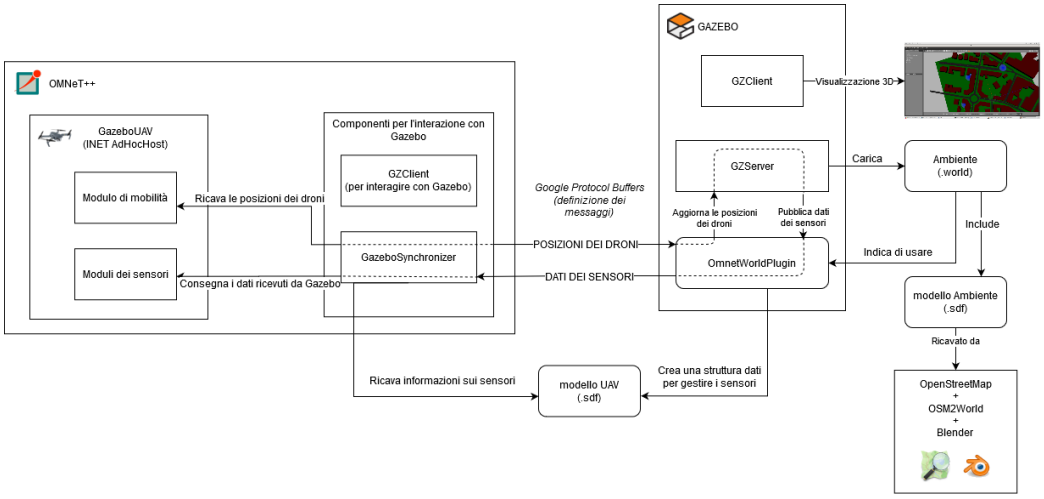


Figura 3.4: Architettura del sistema integrato. Le linee tratteggiate mostrano il flusso di esecuzione da OMNeT++ verso Gazebo, che comincia con il recupero dello posizioni dei droni dal loro modulo di mobilità.

Componenti del sistema:

- GazeboSynchronizer: un modulo in OMNeT++ che si occupa della comunicazione con Gazebo
- OmnetWorldPlugin: un plugin che si occupa della comunicazione con OMNeT++
- I file che definiscono la struttura dei messaggi utilizzati nella comunicazione tra i due simulatori
- Il modulo dei droni in OMNeT++ (GazeboUAV in figura 3.4), derivante dal modello AdHocHost dell'INET framework, che contiene tutti i droni della simulazione. Ogni drone ha due componenti importanti per il sistema:
  - ◇ I moduli dei sensori, dove conservare i dati dei sensori ricevuti da Gazebo
  - ◇ Il modulo di mobilità, da cui ricavare la posizione del drone da passare a Gazebo

- gzclient, che permette a OMNeT++ di comunicare con Gazebo sfruttando il sistema di comunicazione di quest'ultimo, basato sui topic

- il gruppo di applicazioni composto da OpenStreetMap, OSM2World e Blender, che vengono usati per ricavare ambienti realistici dove effettuare la simulazione

- Gazebo (gzclient + gzserver): il primo che che permette di effettuare la simulazione nell'ambiente specificato, utilizzando vari sensori, e il secondo che permette di visualizzare lo stato della simulazione in ambiente 3D

Gli elementi principali per la realizzazione del sistema integrato sono GazeboSynchronizer e OmnetWorldPlugin.

Il GazeboSynchronizer è un modulo di OMNeT++ che viene aggiunto alla file .ned della simulazione e va a interagire con il modulo dei droni lì presente: dei droni andrà a interagire con il modulo di mobilità, per recuperare l'informazione sulle posizioni, e con i moduli dei sensori, per depositare i dati dei sensori ricevuti da Gazebo.

OmnetWorldPlugin è un pezzo di codice che si può indicare in maniera molto semplice a Gazebo di utilizzare: basta inserire un suo riferimento all'interno del file .world in cui si vuole usare. Quando il file .world viene caricato da Gazebo, il plugin viene utilizzato per aggiungere le funzionalità necessarie per l'integrazione.

Questi e altri elementi di OMNeT++ e Gazebo verranno descritti meglio più avanti, dopo aver descritto il modo in cui i due sistemi comunicano.

3.3.3 Comunicazione e inizializzazione

Proseguendo il discorso sul punto di contatto tra OMNeT++ e Gazebo, c'è da dire come questi due sistemi comunicano tra di loro. **Il contatto tra i due sistemi avviene sfruttando il meccanismo di comunicazione presente in Gazebo, cioè il meccanismo publish/subscribe dei topic**.

- Subscribe - si indica una funzione di callback, che viene eseguita ogni volta che viene ricevuto un messaggio su quel topic
- Publish - pubblicazione di un messaggio su un topic (che provoca l'esecuzione delle funzioni di callback di chi ha fatto subscribe)

I due simulatori fanno publish di un messaggio quando il loro turno è finito, mentre riprendono l'esecuzione nel momento in cui ricevono un messaggio sul topic su cui hanno fatto subscribe.

In particolare, OMNeT++ dopo aver fatto publish per avvertire a Gazebo di agire, resta in attesa tramite un comando di *sleep* presente in un ciclo, dal quale si esce solamente una volta ricevuta risposta da Gazebo, sempre tramite i topic.

I topic utilizzati sono tre:

- *setup* - utilizzato solo per la fase di inizializzazione
- *new_event* - per indicare a Gazebo di eseguire un passo di simulazione
- *gazebo_response* - per inviare messaggi a OMNeT++

Mentre i topic *new_event* e *gazebo_response* vengono utilizzati continuamente nel corso della simulazione, il topic *setup* viene utilizzato solo all'inizio, quando ancora la simulazione non è partita.

La procedura di inizializzazione è la seguente:

- avvio di Gazebo con l'indicazione del file .world da usare (che provoca il caricamento del plugin per l'integrazione con OMNeT++)
- inizializzazione della simulazione in OMNeT++ (non il comando per farla partire ma per prepararla all'avvio)

Una volta eseguito questi due passi i modelli dei droni vengono caricati in Gazebo e la simulazione può partire. Praticamente, la fase di inizializzazione

serve per indicare a Gazebo quali sono i modelli da caricare e i sensori da gestire.

È importante che Gazebo sia già in esecuzione al momento della fase di inizializzazione della simulazione di OMNeT++, in quanto il modulo GazeboSynchronizer, in fase di inizializzazione, effettua una publish sul topic *setup*, provocando quindi un blocco nel codice tramite *sleep* come descritto precedentemente.

3.3.4 Struttura dei messaggi

I messaggi scambiati tra OMNeT++ e Gazebo sono definiti usando google Protocol Buffers (protobuf), che è il sistema utilizzato da Gazebo. I messaggi vengono definiti in un file, elencando i valori che vi fanno parte; si indica il tipo del campo, se contiene solo un valore o una lista, se è opzionale o obbligatorio.

Dato che Gazebo utilizza questo sistema esistono già diversi messaggi definiti, per esempio quelli che contengono i dati dei sensori. Infatti, per ottenere i dati aggiornati da un sensore, quello che si fa é una subscribe sul topic di quel sensore.

Questo messaggio contiene le informazioni che devono essere passate a OMNeT++. Quello che si fa é consegnare questi messaggi raggruppati per tipo di sensore e per drone, sfruttando il fatto che i messaggi possono contenere in essi altri messaggi.

Il messaggio con i dati dei sensori è così composto:

- lista di droni
  - ◇ nome del drone
  - ◇ lista di sensori di prossimità
    - ○ nome del sensore
    - ○ dati del sensore (il messaggio restituito dal sensore sul suo topic)

- ◇ ...altre liste di sensori come sopra...

Il messaggio che da OMNeT++ viene inviato a Gazebo, contenente le posizioni dei droni, è invece così composto:

- tempo di simulazione (opzionale)
- lista di droni
  - ◇ il nome del drone
  - ◇ la posizione del drone
  - ◇ il nome del file del modello utilizzato

Il tempo di simulazione serve per indicare a Gazebo quanto tempo andare avanti nella simulazione prima di cedere il passo a OMNeT++, ma al momento non viene utilizzato (il passo di simulazione è stato fissato a un certo valore). Sia il tempo di simulazione che la posizione del drone sono due messaggi predefiniti di Gazebo. Il nome del file del modello ènecessario solo nel messaggio di inizializzazione.

3.3.5 Lato Gazebo

La parte di integrazione con OMNeT++ è contenuta in un World plugin, che viene caricato tramite riferimento dei file .world e permette di controllare la simulazione. La prima cosa che fa questo plugin è mettere in pausa la simulazione, perchè di base questa inizia non appena Gazebo si è avviato; il comando di pausa avviene nel metodo *Init()*. In seguito viene chiamato il metodo *Load(..)*, dove si fa subscribe dei topic *setup* e *new_event*.
Una cosa importante che deve fare il plugin è gestire i sensori, ovvero fare subscribe sui topic dei sensori per riceverne i dati e quindi inviarli a OMNeT++. Il problema è, una volta mandata avanti la simulazione del tempo necessario, capire se dei sensori stanno per generare nuovi dati, perchè una volta mandata avanti la simulazione i dati potrebbero non essere ancora disponibili.I sensori hanno un parametro chiamato updateRate, che indica il

numero di aggiornamenti che quel sensore effettua in un secondo. Quello che si fa, dopo essere avanzati di un passo di simulazione, è usare questo dato e il tempo di ultimo aggiornamento del sensore per capire quanti sensori stanno per generare nuovi dati.
I dati dei sensori devono anche essere organizzati in una certa maniera nel messaggio da inviare a OMNeT++ (vedi la sottosezione 3.3.4); per fare questo **viene creata una struttura dati per i sensori**, che viene popolata durante la fase di inizializzazione. In questa struttura dati, delle classi che si occupano di ricevere e memorizzare i dati dei sensori sono organizzate per drone, ovvero esistono una sorta di liste di sensori per ogni drone (vedi figura 3.5).
Una volta terminato il passo di simulazione, si controlla quanti sensori si devono aggiornare: se nessun sensore deve aggiornarsi viene subito spedito a OMNeT++ un messaggio "vuoto" (cioè privo di dati dei sensori) per indicare che Gazebo ha finito la sua parte di simulazione, altrimenti è necessario attendere che tutti i sensori vengano aggiornati e poi prelevarne i dati. Quando un sensore si aggiorna, il gestore del sensore corrispondente ne riceve il nuovo dato (vedi figura 3.5) e notifica un componente del plugin dell'evento: il plugin, una volta ricevuto notifica da tutti i sensori che si dovevano aggiornare, costruisce il messaggio per OMNeT++ con i dati dei sensori ed esegue la publish.

Importante: Riguardo all'operazione di publish, c'è da dire che di base non invia subito il messaggio, ma attende un certo quantitativo di tempo. È però possibile indicare alla publish di inviare subito il messaggio (cosa che viene fatta nel plugin) risparmiando tempo, ma comunque si è limitati dal tempo di attesa dei dati dei sensori, che vengono inviati tramite publish con tempo di attesa. Per questo, quando è il momento dell'aggiornamento dei sensori, Gazebo impiega molto più tempo prima di ritornare il controllo a OMNeT++, perchè deve aspettare che le publish non istantanee dei sensori ritornino i dati da comunicare a OMNeT++.

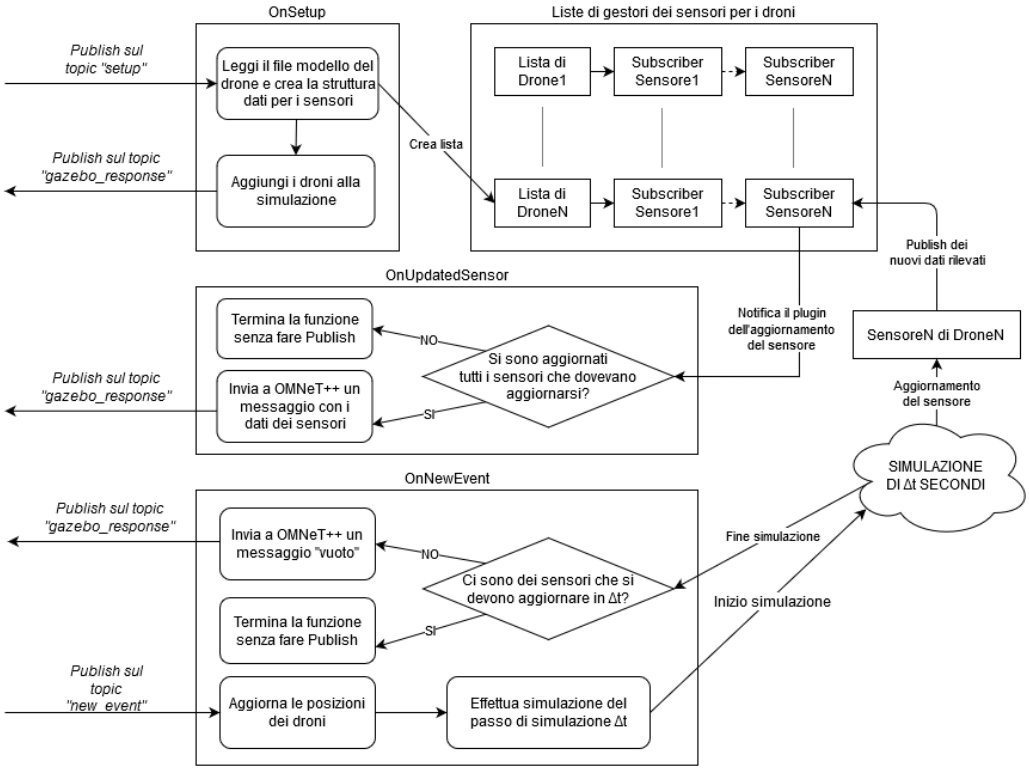


Figura 3.5: La gestione dei sensori dei droni lato Gazebo. A sinistra sono riportate operazioni di publish tramite cui Gazebo e OMNeT++ interagiscono. In alto si vede l'operazione di inizializzazione (topic *setup*), mentre in basso quella di sincronizzazione (topic *new_event*).

3.3.6 Lato OMNeT++

Come detto precedentemente, un modulo apposito viene aggiunto al file .ned per gestire il rapporto con Gazebo. Questo modulo si chiama GazeboSynchronizer, e fa uso di alcuni file header di Gazebo in quanto ha bisogno di utilizzare alcune sue funzioni: i metodi per fare publish e subscribe e le classi per alcuni messaggi.

Durante la fase di inizializzazione, questo modulo fa subscribe sul topic *gazebo_response* e invia il messaggio di inizializzazione a Gazebo. Il messaggio di inizializzazione in realtà non è diverso dai messaggi che poi il modulo invierà in seguito.

Prima di parlare di alcune operazioni che fa questo modulo, c'è però prima bisogno di spiegare come è strutturata la simulazione e come sono strutturati i droni.

I componenti della simulazione

Figura 3.6: File .ned della simulazione

L'immagine nella figura 3.6 mostra il file .ned di OMNeT++ della simulazione. I componenti principali sono GazeboSynchronizer e il modulo vettore drones. GazeboSynchronizer comunica con i droni del vettore drones per prelevare la posizione dei droni e per consegnare i dati dei sensori, ricevuti da Gazebo. Mentre durante la prima fase di programmazione per i droni è stato usato un modulo semplice, più avanti sono stati sostituiti da un modulo (chiamato GazeboUAV) che va ad estendere il modulo AdHocHost dell'INET framework.

Quello nell'immagine (Fig. 3.7) è il file .ned del modulo GazeboUAV, che va ad aggiungere poche cose al modulo che estende, principalmente il vettore di sensori, dove verranno aggiunti i moduli dei sensori che conterranno i dati

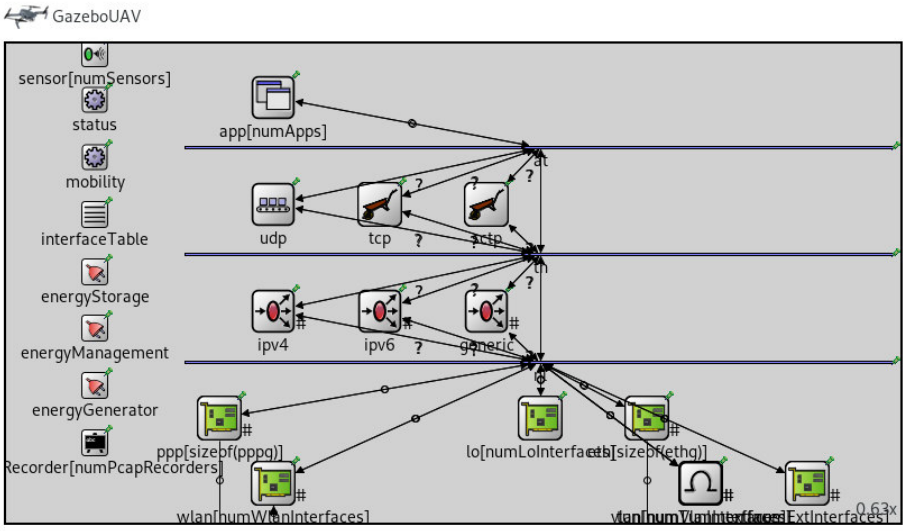


Figura 3.7: File .ned del modulo GazeboUAV

ricevuti da Gazebo. Il GazeboSynchronizer andrà quindi a inserire i dati nei moduli di questo vettore.

Un'altro elemento di interesse presente nell'immagine, che fa parte di un modulo base di inet, è il modulo di mobilità: questo è il modulo responsabile dei movimenti del drone ed è questo modulo che il GazeboSynchronizer va ad interrogare per ricavare la posizione del drone.

Il modulo dei sensori

Il drone che verrà utilizzato nella simulazione avrà un certo numero di sensori, e ogni sensore presente nel drone verrà rappresentato da un modulo del tipo corrispondente che andrà a far parte del vettore di sensori visto sopra. I sensori sono dei moduli molto semplici, in quanto il loro unico scopo è quello di memorizzare l'informazione del sensore ricevuta da Gazebo. Oltre a questa informazione possiedono solamente il nome, che permette di identificarli. Ogni tipo di sensore ha la sua classe modulo e il suo file .ned, ma tutti ereditano da una classe base comune (vedi figura 3.8).

Il GazeboSynchronizer si occupa di popolare il modulo dei sensori in fase di inizializzazione, e lo fa andando a leggere il file del modello del drone, dove

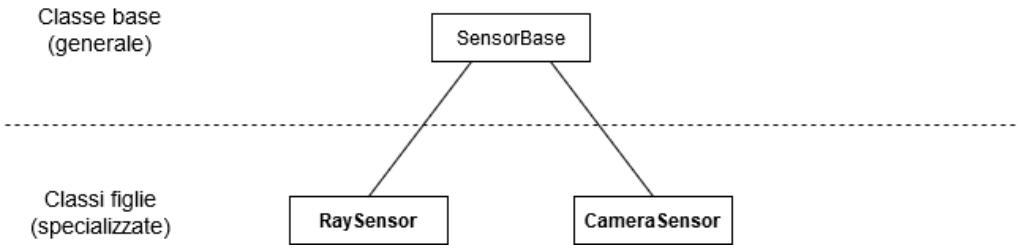


Figura 3.8: Ereditarietá dei moduli dei sensori. Le classi dei sensori che vengono effettivamente utilizzate derivano da una classe base comune.

è contenuta l'informazione sui sensori in esso presenti. Quando poi lo stesso modulo riceve il messaggio da parte di Gazebo, controlla se ci sono dati dei sensori e provvede a consegnarli al modulo del sensore corrispondente, confrontando tipo e nome del sensore.

Capitolo 4

Caso di studio

In questo capitolo viene descritto il caso di studio che si è voluto simulare tramite il sistema integrato realizzato. Si parla quindi della logica della mobilità utilizzata, basata sulle molle virtuali, e di cosa si è fatto per realizzare la simulazione.

4.1 Introduzione

Avendo realizzato il sistema integrato, si vuole studiare attraverso questo sistema un certo tipo di mobilità. Come descritto all'inizio del documento, si è voluto riportare nella simulazione un tipo di mobilità di gruppo basato sulle molle. In questa mobilità i droni decidono come muoversi sulla base dei messaggi ricevuti dai droni vicini: a seconda della potenza dei segnali ricevuti (e quindi della vicinanza agli altri droni), e di un certo requisito di qualità del servizio, ci si allontana/avvicina rispetto altri droni fino a che non si raggiunge un equilibrio.

In questo modo i droni si posizionano in modo da formare una rete che punta a fornire un certo livello di qualità del servizio.

Oltre a questo, però, si vuole osservare il comportamento della rete di droni con l'aggiunta di un altro tipo di mobilità: si vuole inserire un drone leader,

ovvero un drone con un suo percorso prefissato il cui movimento influenza gli altri droni, in quanto anche lui, come gli altri droni, manda messaggi in broadcast e questo provoca nei droni dello sciame la creazione delle molle virtuali, che quindi spingono questi droni a seguire il leader.

Si vuole quindi vedere come lo sciame di droni segue il proprio leader.

In aggiunta a ciò, si vuole applicare il sistema delle molle anche come protezione dagli ostacoli. I droni vengono equipaggiati con dei sensori di prossimità, 8 sensori distribuiti attorno ad essi, con i quali possono rilevare gli ostacoli nelle vicinanze e generare molle che li allontanino da essi.

Mettendo in gioco questi sensori, diventa necessario ottenere un ambiente con ostacoli dove condurre la simulazione; per avere questo si è sfruttato OpenStreetMap per generare il modello di una città da inserire in Gazebo.

Il sistema integrato OMNeT++ & Gazebo permette di effettuare simulazioni di scenari come quello appena descritto in maniera conforme alle necessità, offrendo:

- gestione della rete di comunicazione: fornita da OMNeT++ e dall'INET framework
- gestione dei sensori: fornita da Gazebo
- riproduzione dell'ambiente di simulazione: attraverso l'uso di OpenStreetMap e Gazebo

Il sistema OMNeT++ & Gazebo è quindi uno strumento adatto ad effettuare simulazioni di sistemi multi-agente a mobilità autonoma che si vuole condurre in un ambiente realistico. Al contrario, riprodurre una simulazione di questo tipo mediante un unico simulatore non fornirebbe in maniera adeguata tutte e tre le funzionalità elencate, portando a una simulazione meno precisa.

4.2 Molle dello sciame

Iniziamo con il parlare di come queste molle virtuali decidono come il drone deve muoversi nell'ambiente. Come detto prima, questo tipo di mobilità punta a posizionare i droni in maniera da formare una rete che fornisca una certa qualità del servizio richiesta, espressa tramite un parametro *targetLinkBudget*. Il Link Budget di una comunicazione wireless è la differenza tra potenza del segnale ricevuto e sensibilità dell'antenna ricevente; maggiore il valore del Link Budget, minori gli eventuali disturbi nel segnale che possono interferire nel corretto funzionamento del servizio che fa uso della connessione tra i due nodi.

Il *targetLinkBudget* indica quindi la qualità del servizio richiesta e porta i droni a spostarsi in modo da raggiungere tale requisito: le molle infatti spingono i droni a muoversi per raggiungere il valore di questo parametro.

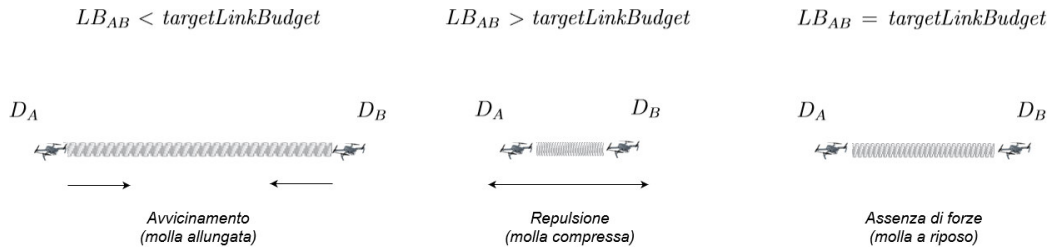


Figura 4.1: La forza che la molla virtuale esercita sui droni a seconda del Link Budget.

Siano D_A e D_B due droni. Quando D_A riceve un messaggio da D_B, quello che succede è questo:

- D_A esamina la potenza del segnale del messaggio ricevuto e calcola il link budget (LB_{AB})
- D_A ottiene la posizione di D_B e in base a questo crea/aggiorna la molla
  - ◇ se $LB_{AB} < targetLinkBudget$, la molla spinge D_A verso D_B
  - ◇ se $LB_{AB} > targetLinkBudget$, la molla allontana D_A da D_B

- ◇ se $LB_{AB} = targetLinkBudget$, la molla non influenza lo spostamento di D_A

Ogni drone utilizza il GPS per sapere la propria posizione e invia tale posizione agli altri droni nel suo messaggio broadcast, in quanto necessaria per la creazione della molla. Infatti, la direzione della spinta dipende dalla posizione dei due droni, e spinge ad andare incontro all'altro o ad allontanarsene muovendosi in direzione opposta. La forza della molla, e quindi la velocità con cui il drone si muove, è data invece dalla differenza tra LB_{AB} e *targetLinkBudget*. La formula utilizzata è:

$$F_{AB} = k * \frac{\max(LB_{AB}, targetLinkBudget)}{\min(LB_{AB}, targetLinkBudget)} \tag{4.1}$$

dove F_{AB} è la forza della molla e k un valore che indica quanto la molla è rigida. k può essere variato per rendere i droni più o meno sensibili/reattivi.

Per ogni drone di cui D_A riceve il messaggio, questo crea una nuova molla, che partecipa alla decisione su come muoversi. Lo spostamento generato dalla molla è dato dal seguente vettore:

$$S_{AB} = \text{vettore\_direzione} * F_{AB} \tag{4.2}$$

Sia M_A l'insieme dei vettori velocità generati dalle molle del drone D_A(i vettori generati dalla formula 4.2); il movimento del drone è determinato dalla somma dei vettori velocità:

$$S_A = \sum_{s \in M_A} s \tag{4.3}$$

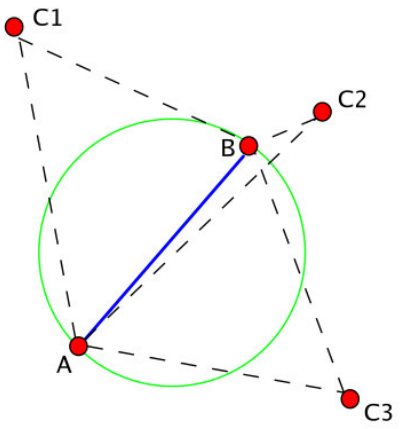
4.2.1 Generazione e rimozione delle molle

Precedentemente si è definito il funzionamento delle molle, cioè la forza e quindi lo spostamento da esse generato. Lo spostamento e il posizionamento dei droni dipende però anche dalla logica di creazione e rimozione delle molle, perchè possono esserci diversi modi di collegare tra di loro i droni con le molle

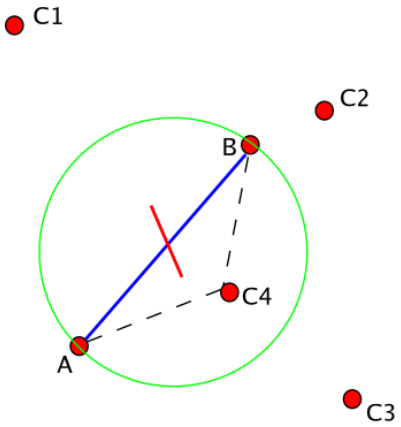
(come si vede in [17]). Il metodo usato in questo progetto è quello dell'angolo acuto.

Quando un drone riceve il messaggio di un altro drone calcola il Link Budget e ricava la posizione di quel drone. Quindi, se il messaggio arriva da un drone con cui non si è ancora collegati tramite molla, utilizza il test dell'angolo acuto per decidere se creare o no la molla. Siano D_B il drone che invia il messaggio e D_A il drone che riceve il messaggio. Il controllo eseguito da D_A è il seguente:

- Per ogni drone D_C con cui D_A è collegato tramite molla, D_A effettua il test dell'angolo acuto usando le posizioni di D_A, D_B e D_C
  - ◇ Siano A,B e C le posizioni dei droni sopra citati. Viene calcolato l'angolo $A\widehat{C}B$
  - ◇ Se $A\widehat{C}B > 90°$, l'angolo è ottuso ed il test non è superato
  - ◇ Se $0°< A\widehat{C}B \leq 90°$, l'angolo è acuto ed il test è superato
  - ◇ Se $A\widehat{C}B = 0°$(le tre posizioni sono allineate), si effettua un controllo aggiuntivo..
    - · Sia XY la distanza tra i droni D_X e D_Y.
    - · Se $CA < BA \wedge CB < BA$, allora il test non è superato
    - · In caso contrario, il test è stato superato
- La molla viene creata solo se, per ogni drone D_C, il test è stato superato



(a) Test superato per tutti i droni C_X, creazione della molla con B



(b) Test non superato ($A\widehat{C_4}B$ è ottuso), la molla non viene creata

Figura 4.2: Test dell'angolo acuto su drone A al momento della ricezione di un messaggio dal drone B. Immagini tratte da [17].

Questo test evita di creare molle con i droni presenti in una certa direzione quando si ha già almeno una molla con un drone più vicino all'incirca in quella direzione. Si possono vedere degli esempi nelle immagini in figura 4.2. È un test basato solo sulla posizione, non sulla forza della molla.

Figura 4.3: Esempio di formazione molle tra tre droni più o meno allineati

Facendo un altro esempio, se ci sono 3 droni più o meno allineati, i droni

agli estremi formeranno un collegamento solo con il drone centrale, anche se ricevono i messaggi da quello dall'altra parte (fig. 4.3). Il drone centrale, invece, formerà le molle con entrambi i droni.

Le molle, però, una volta create, potrebbe non essere il caso di mantenerle in eterno. Ci sono due casi nel quale le molle vengono eliminate:

- se non si ricevono più messaggi dal drone associato da un certo ammontare di tempo

- se il drone associato non supera più il test dell'angolo acuto (perchè ci sono stati spostamenti o aggiunte di altre molle)

Quello che si fa è eseguire periodicamente un controllo per vedere se la molla deve essere eliminata, perchè una di queste due condizioni si è verificata.

4.3 Molle per evitare gli ostacoli

Le molle che hanno l'obiettivo di spingere i droni lontano dagli ostacoli sono simili alle molle dello sciame: come le molle prima citate hanno una direzione, una forza e un parametro che ne determina la rigidità. Diversamente dalle molle precedenti, che vengono create per ogni drone nelle vicinanze, queste molle sono associate a dei sensori, una molla per ogni sensore.

4.3.1 I sensori

Il drone che viene usato nella simulazione è equipaggiato con 8 sensori di prossimità (vedi figura 4.4), uno per ogni punto cardinale (Nord, Sud, Ovest, Est) e per i punti a loro intermedi (Nord-Est, Nord-Ovest, Sud-Est, Sud-Ovest). Ogni sensore ha quindi un raggio di percezione ostacoli ampio 45°, e percepisce ostacoli fino a una certa distanza (nella simulazione fatta i sensori percepiscono ostacoli entro 3 metri).

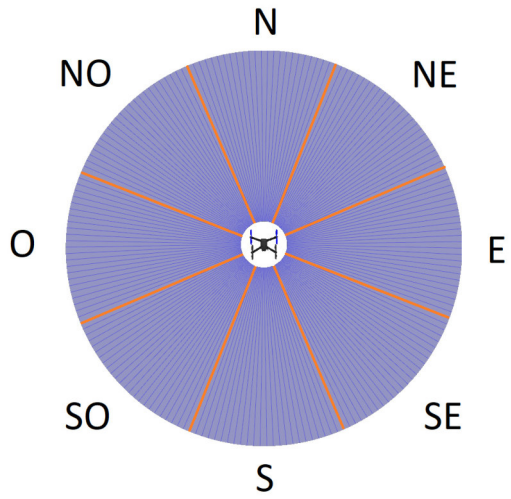


Figura 4.4: Gli 8 sensori dei droni, nominati secondo i punti cardinali prendendo come nord la parte frontale del drone.

4.3.2 Spostamento dovuto ai sensori

Dato che il drone deve allontanarsi dagli ostacoli, quando un ostacolo viene rilevato da uno dei sensori viene generata una molla che spinge il drone dalla parte opposta rispetto alla posizione del sensore (per esempio, se il sensore di Nord-Ovest percepisce un ostacolo si genera una forza che spinge il drone in direzione Sud-Est).

La forza della molla dipende dalla distanza dell'ostacolo dal drone. Sia d la distanza dal sensore e k la rigidità della molla. La forza generata dalla molla è data da:

$$F = k * (10/d) \tag{4.4}$$

Nella simulazione il valore usato per k è stato 0.4. In maniera simile al calcolo dello spostamento del drone tramite le molle dello sciame (formula 4.3) lo spostamento generato dai sensori di prossimità è dato dalla somma dei vettori velocità da loro generati.

4.3.3 Molle sciame + Molle ostacoli

Usando sia le molle dello sciame che le molle per evitare gli ostacoli, la formula finale per il movimento del drone prende in considerazione tutte le molle generate. Dati MS_A le molle dello sciame e MO_A le molle degli ostacoli generate dal drone D_A, lo spostamento del drone è ricavato sommando tutti i vettori velocità delle molle:

$$S_A = \sum_{ms \in MS_A} ms + \sum_{mo \in MO_A} mo \tag{4.5}$$

4.4 Realizzazione

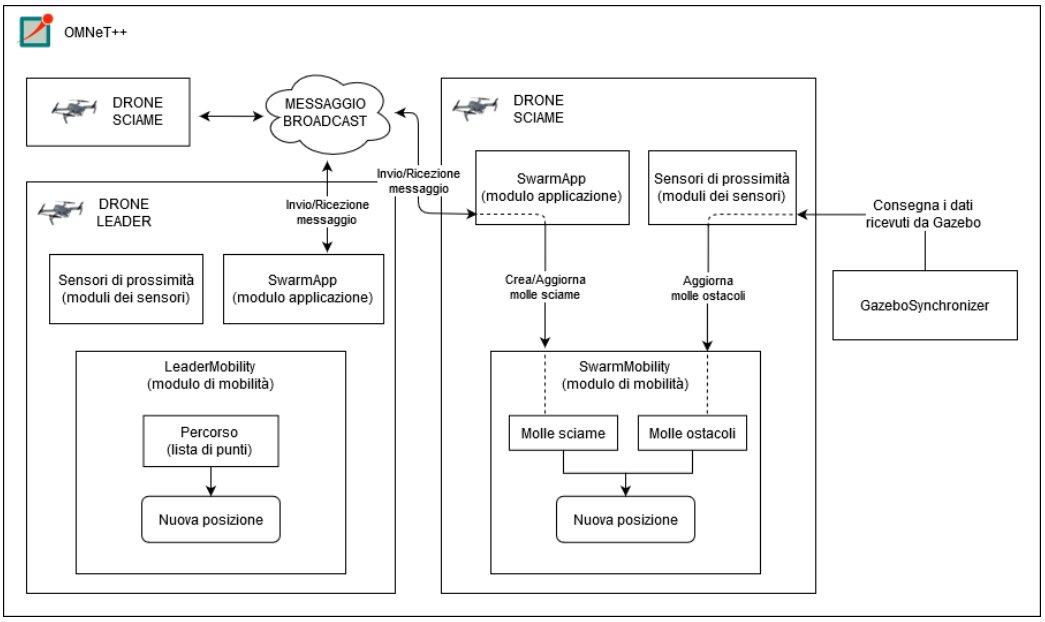


Figura 4.5: Schema di realizzazione del caso di studio. I droni si scambiano messaggi tramite il modulo applicazione. Su ricezione di un messaggio o dei dati di un sensori, il modulo di mobilità aggiorna la molla corrispondente.

Gli 8 sensori di prossimità sono stati inseriti nel file .model del drone. Ognuno di loro è uguale a parte per la direzione e per il nome del sensore, che fa riferimento ad essa (N per Nord, NE per Nord-Est, ecc.). OMNeT++ (il

modulo *GazeboSynchronizer*) provvederà a creare i moduli per questi sensori in fase di inizializzazione.

Il codice che determina la posizione dei droni si trova in OMNeT++, precisamente nel modulo di mobilità dei droni (vedi sezione 3.3.6 - Lato OMNeT++). Sono stati creati i moduli di mobilità per il drone leader e per i droni dello sciame (vedi figura 4.5).

Il modulo di mobilità del leader contiene una lista di coordinate. Nella funzione *move* del modulo, che viene chiamata quando la posizione del drone deve essere aggiornata, semplicemente il drone si muove di una velocità fissa verso il prossimo punto della lista.

Per quanto riguarda il modulo di mobilità che fa uso delle molle, nella funzione *move* viene indicato di muoversi secondo la formula 4.5, cioè sommando i vettori velocità di tutte le molle.

4.4.1 Creazione e aggiornamento molle

Come si aggiornano le molle?

Le funzioni per creare e aggiornare le molle (sciame e sensori) sono tutte contenute nel modulo di mobilità, ma sono chiamate da due elementi diversi. Infatti, le molle dei sensori si aggiornano quando si ha un aggiornamento dei sensori, mentre le molle per lo sciame si aggiornano quando si riceve un messaggio da un altro drone (vedi figura 4.5).

Per quanto riguarda le molle dei sensori, gli aggiornamenti dei sensori provengono da Gazebo, ed è il modulo *GazeboSynchronizer* a gestire la comunicazione con Gazebo; per questo, dopo aver depositato i dati dei sensori nei moduli dei sensori, è il modulo *GazeboSynchronizer* ad avvertire i moduli di mobilità dei droni di aggiornare le molle dei sensori.

Mentre le molle dei sensori vengono create dal modulo di mobilità in fase di inizializzazione, le molle per la mobilità dello sciame vengono create e aggiornate da un altro modulo, il modulo applicazione.

È stato creato un modulo di tipo applicazione chiamato *SwarmApp*, che si occupa di:

- inviare periodicamente messaggi in broadcast
- ricevere messaggi e creare la molla

Quello che l'applicazione fa una volta ricevuto un messaggio è calcolare il link budget, ricavare la posizione del drone da cui è arrivato il messaggio, chiamare la funzione del modulo di mobilità per l'aggiornamento della molla passandogli i precedenti parametri. Il modulo di mobilità aggiorna la molla, o la crea se questa ancora non esiste.

4.5 OpenStreetMap

Per effettuare la simulazione è stato necessario ottenere un ambiente con ostacoli dove poter testare la mobilità dei droni. Si è ricavato questo ambiente utilizzando come fonte dati OpenStreetMap, che fornisce delle mappe del mondo reale online libere da usare.

OpenStreetMap è un progetto collaborativo per la creazione di mappe, a cui ogni persona può dare un contributo, come per esempio avviene in Wikipedia.

Andando sul sito di OpenStreetMap, si è fatto download dei dati relativi a un'area geografica tramite la funzionalità *Export*, e si sono poi seguiti i passi indicati nel sito web in [29] per la creazione di un file .model dell'area da poter utilizzare in Gazebo (figura 4.6).

La possibilità di ricavare l'ambiente dove condurre simulazioni sfruttando le informazioni di OpenStreetMap è un notevole vantaggio, in quanto permette velocemente di ottenere un ambiente ricavato da mappe reali. È quindi possibile effettuare la propria simulazione in un ambiente realistico e, in alcuni casi, testare il proprio sistema nei luoghi dove effettivamente verrà utilizzato senza doversi preoccupare di creare personalmente l'ambiente, risparmiando tempo.

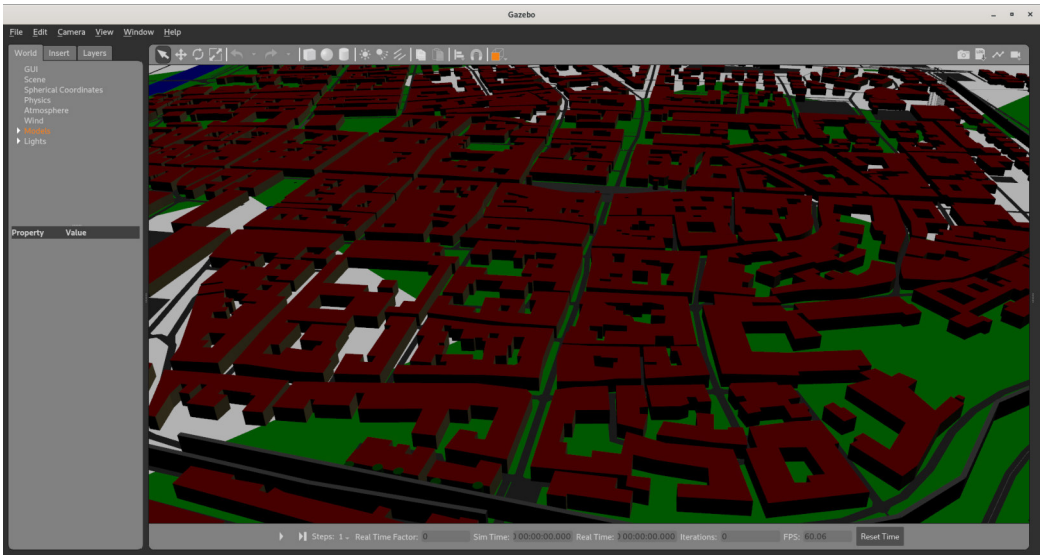


Figura 4.6: Il modello della città utilizzato nelle simulazioni e ricavato da OpenStreetMap.

Capitolo 5

Valutazioni e Performance

In questo capitolo vengono descritti i test che sono stati fatti per valutare il sistema creato e la mobilità dello sciame basata su molle. I risultati dei test sono mostrati e presentati mediante vari grafici.

5.1 Scenario

I test sono stati effettuati per due diversi scopi:

- valutare il sistema OMNeT++ & Gazebo
- osservare il comportamento dei droni e valutare il sistema delle molle in diversi scenari

Per quanto riguarda la valutazione del sistema, si è pensato di osservare i seguenti valori:

- tempo di risposta di Gazebo (o attesa di OMNeT++) al variare del numero dei sensori su un singolo drone
- tempo di risposta di Gazebo al variare del numero di droni
- differenza del rapporto $\frac{SimSec}{RealSec}$ (secondi simulati per ogni secondo reale) tra il sistema creato e il singolo OMNeT++ senza l'utilizzo dei sensori

I primi due valori (i tempi di risposta di Gazebo) servono per verificare l'influenza che il numero dei sensori e il numero dei droni ha sul tempo necessario a Gazebo per completare la sua parte della simulazione, e quindi per capire le possibilità di Gazebo, mentre l'ultimo è per verificare il ritardo totale che Gazebo aggiunge alla simulazione condivisa, mettendo quindi a confronto il solo OMNeT++ con il sistema integrato.

Per quanto riguarda invece lo studio dei droni, si è inizialmente pensato di effettuare test in 3 scenari diversi:

- campo aperto (privo di ostacoli)
- zona con ostacoli piccoli e sparsi
- zona con molti ostacoli (es. città)

Durante i test si è posto maggiore interesse nell'ambiente con molti ostacoli, che è stato ricavato tramite OpenStreetMap. Nella figura 5.1 si vede il modello usato per i test, che rappresenta una parte della città di Rimini. Sono inoltre segnati i percorsi utilizzati nei test.

Nei test effettuati per lo studio dei droni si è esaminato principalmente il comportamento dei droni per ciò che si poteva notare attraverso l'ambiente grafico di Gazebo, andando poi a vedere alcuni valori nelle statistiche registrate in OMNeT++. In particolare, si è andato a osservare il Link Budget (LB) medio dello sciame, cioè la media dei Link Budget dei droni tra loro collegati (che hanno una molla che li lega).

I test sono stati ripetuti variando alcuni parametri: principalmente il numero di droni, ma anche il targetLinkBudget (quello che le molle mirano a fornire), la forza delle molle, la velocità del leader.

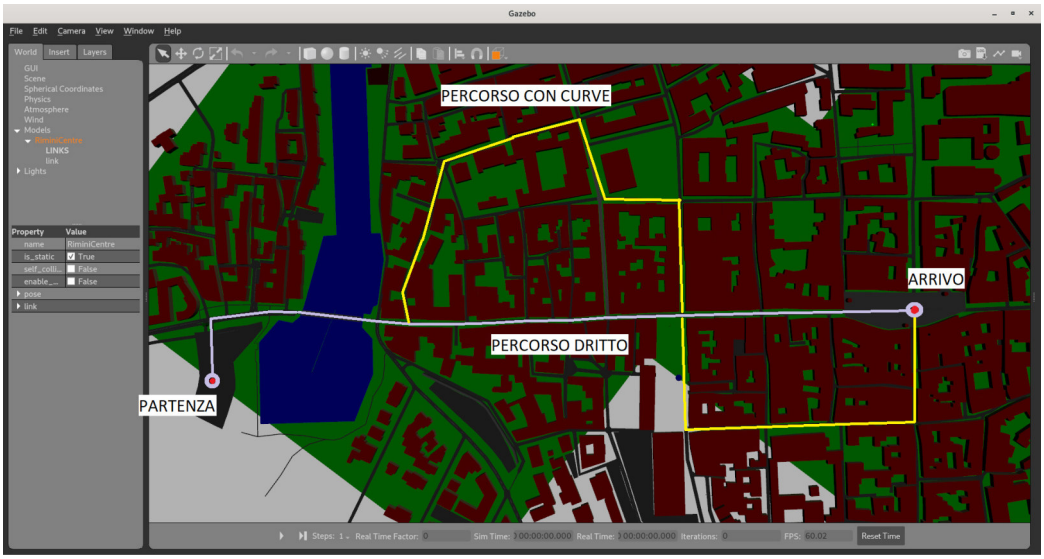


Figura 5.1: Il modello della città utilizzato nelle simulazioni, visto dall'alto. Sono segnati i due percorsi pensati per le simulazioni.

5.1.1 Parametri della simulazione

I parametri base della simulazione, utilizzati dove non diversamente specificato, sono i seguenti:

- numero di droni: 8
- numero di sensori per drone: 8
- targetLinkBudget: 30 dBm (per avere un'alta qualità del servizio)
- portata dei sensori: 3 m
- velocità del drone leader: 3 m/s
- parametro k delle molle tra due droni: 10
- parametro k delle molle per gli ostacoli: 0.4

I parametri invece mai cambiati, sono:

- altezza di volo dei droni: 2 m
- velocità massima dei droni sciame: 5 m/s
- transmitter power: 0.02 W
- receiver sensitivity: -85 dBm
- parametro alpha del sottomodulo pathLoss del modulo radioMedium: 3
- intervallo di invio dei messaggi: determinato dalla distribuzione normale troncata (scarto dei valori negativi) con media di 0.5 (s) e deviazione standard di 0.05
- intervallo di aggiornamento delle posizioni dei droni: 0.1 s
- intervallo del controllo di rimozione delle molle dello sciame: 3 s
- durata minima delle molle dello sciame in assenza di segnale dal drone associato: 5 s

Utilizzando come modello di propagazione del modulo radioMedium il FreeSpacePathLoss con valore di alpha sopra citato (3), e i valori di transmitter power (0.02 W), receiver sensitivity (-85 dBm) e targetLinkBudget (30 dBm) per i droni, si ricavano i seguenti dati:

· punto d'equilibrio tra due droni: 8.5 m

· distanza massima di ricezione del messaggio: 85 m

Inoltre, da parte di Gazebo, si sono fatte delle modifiche per adattare il sistema al caso di studio, in modo da aumentare le performance e rendere le simulazioni più veloci. Ogni volta che Gazebo ottiene il suo turno di simulazione, invece di lanciare il comando di simulazione per 0.1 secondi, lo esegue

solamente per il passo minimo, 1 millisecondo. Questo risparmia tempo di simulazione, in quanto per simulare 0.1 secondi Gazebo ci mette circa 0.1 secondi, di base.
Dato che i sensori utilizzano il loro parametro updateRate e il tempo della simulazione per capire quando devono aggiornarsi, con questa modifica diventa necessario modificare il parametro updateRate per fare aggiornare il sensore ogni millisecondo di simulazione invece che 10 volte al secondo; per fare questo basta impostare il parametro a 0, che specifica di aggiornare i sensori a ogni passo di simulazione, che di base equivale a 1 millisecondo.
È possibile indicare, attraverso file .world, il passo di simulazione di Gazebo e la velocità della simulazione (rapporto tempo simulato per tempo reale), ma non è stato testato. La funzione che viene usata per far avanzare la simulazione porta avanti la simulazione di un passo di simulazione; si sarebbe potuto quindi evitare di modificare l'updateRate dei droni e indicare come passo di simulazione 0.1 secondi e come veocità della simulazione 100 ($\frac{secondisimulati}{secondireali}$) per simulare 0.1 secondi in un millisecondo e ottenere lo stesso risultato.

5.2 Risultati

5.2.1 Valutazione del sistema (OMNeT++ & Gazebo)

I test per la valutazione del sistema sono stati effettuati su un computer con Intel® Core™ i7-6500U CPU @ 2.50 GHz e 8 GB RAM. I risultati della figura 5.2 fanno riferimento a un test sull'incidenza del numero dei sensori utilizzati nella simulazione nel tempo di risposta di Gazebo. In questo test si è posto un drone immobile vicino a degli ostacoli, e si è variato ogni volta la quantità dei sensori presenti su di esso (riducendo l'ampiezza dell'angolo di percezione ma mantenendo il numero di campionamenti all'interno di tale angolo).

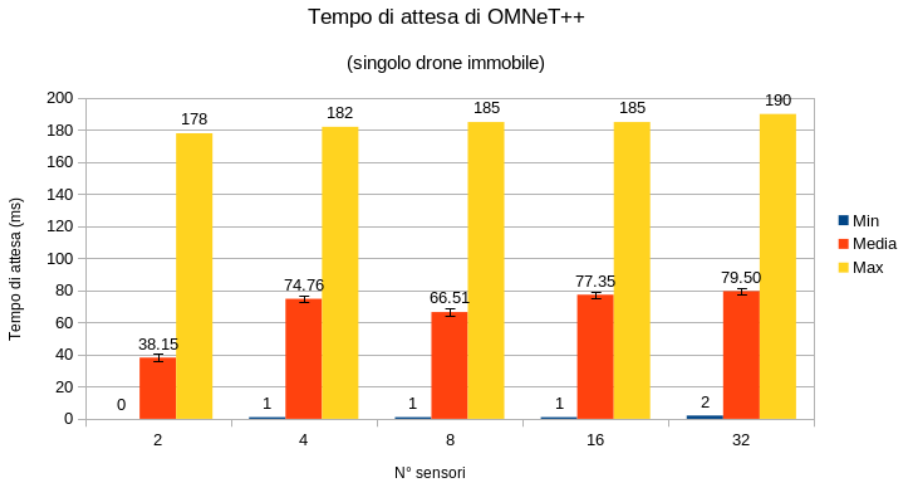


Figura 5.2: Il tempo di sincronizzazione a seconda del numero dei sensori. Il valore medio ha un intervallo di confidenza del 95%.

Nei risultati si vede come i valori siano all'incirca allineati, ad eccezione del tempo medio per 2 soli sensori. Infatti, si vede una notevole differenza nel passaggio da 2 sensori a 4.

Sia dal grafico precedente che dal successivo, si può vedere come il tempo di risposta di Gazebo sia molto variabile. Questo dovrebbe essere dovuto al fatto che Gazebo pubblica i dati del sensore sul topic corrispondente in maniera non bloccante: non viene cioè richiesto un invio immediato del messaggio con i dati, e quindi l'invio può capitare a intervalli di tempo variabili.

Nei successivi test si è lasciato costante il numero di sensori per drone e si è variato il numero di droni. Guardando la figura 5.3 si può vedere come il tempo di risposta con numero di droni tra 2 e 8 non sembra variare molto, mentre con 16 si nota molta differenza, e anche tra 16 e 20 c'è un deciso aumento di tempo medio e minimo. Il tempo massimo mostra di non essere troppo influenzato dal numero di droni.

In seguito si è esaminato la velocità della simulazione in tempo reale impiegato e rapporto tra tempo simulato e tempo reale (SimSec/RealSec); la simulazione dei test consiste nel far muovere i droni nel percorso in città

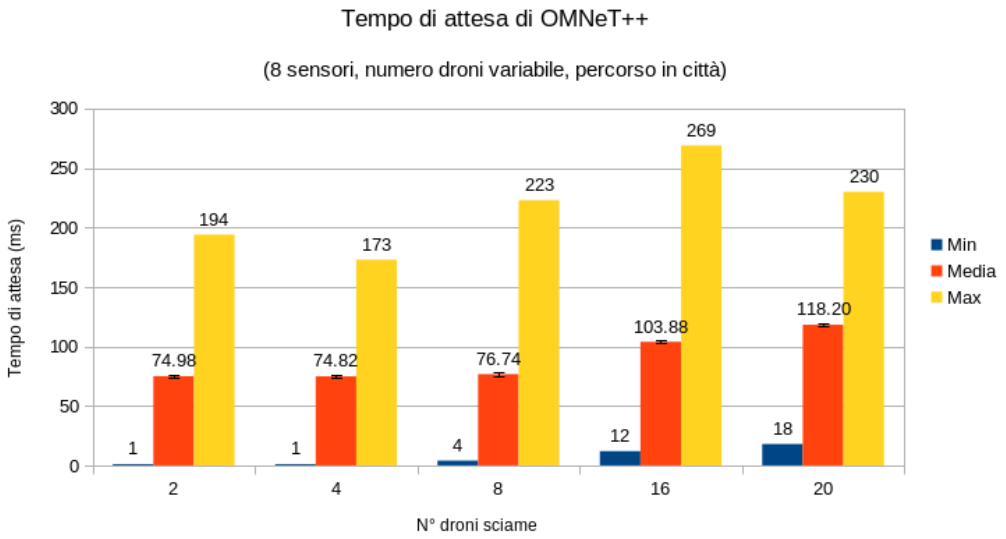


Figura 5.3: Il tempo di sincronizzazione a seconda del numero dei droni, utilizzando 8 sensori per drone. Il valore medio ha un intervallo di confidenza del 95%.

(dritto) per 300 secondi simulati.
Si sono confrontati i tempi del sistema creato con quelli del solo OMNeT++ senza l'uso dei sensori. I risultati sono mostrati nelle figure 5.4 e 5.5.
In figura 5.4 sono mostrati i risultati per la simulazione con uno sciame composto da 8 droni. OMNeT++ da solo è molto più veloce di Gazebo (ci mette $\frac{1}{27}$ del tempo in questo test), e questo mostra quanto ritardo porta l'utilizzo dei sensori.

Se si va però a vedere quello che succede al variare dei numero di droni, si nota come la differenza di tempo tra i due sistemi si riduca con l'aumentare della grandezza della sciame (figura 5.5). Sembrerebbe quindi che l'aumentare del numero di droni, e quindi di invii di messaggi e di controlli per le molle, abbia un grosso impatto sul tempo di lavoro di OMNeT++, riducendo le differenze tra i due simulatori e facendo diventare la gestione dei sensori un carico di lavoro complessivamente di minor impatto.

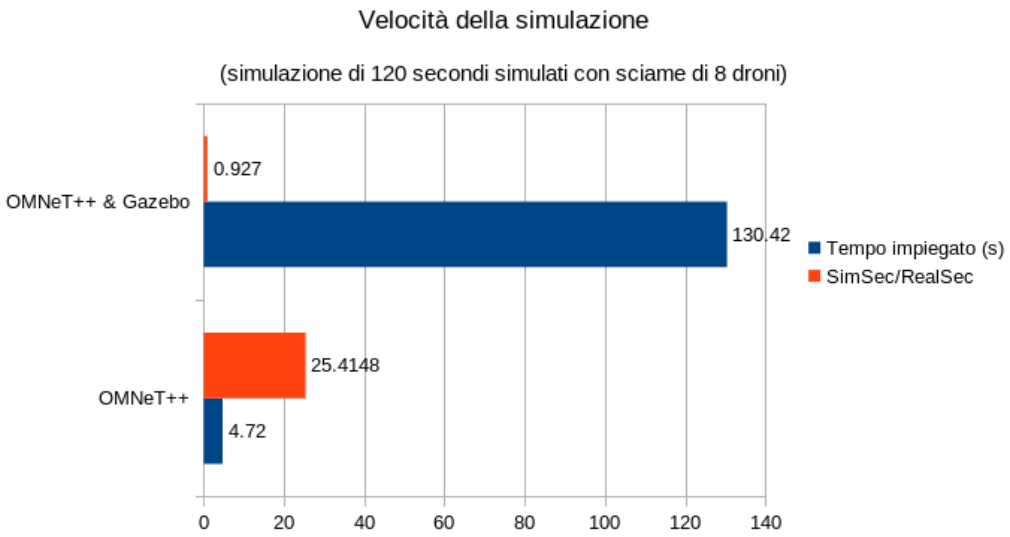


Figura 5.4: La velocità di simulazione a confronto tra con e senza Gazebo, nel caso di 8 droni.

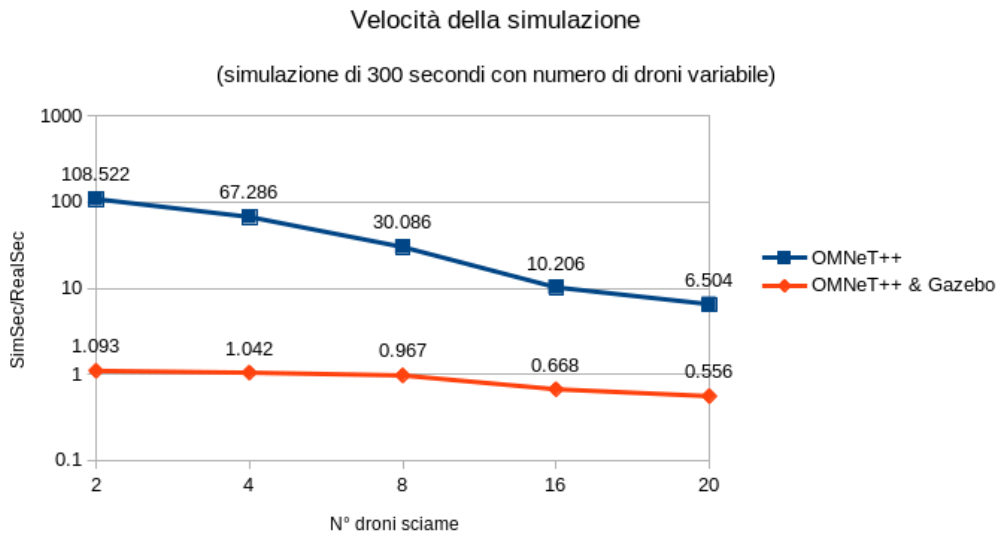


Figura 5.5: La velocità di simulazione a confronto tra con e senza Gazebo, variando il numero di droni (scala logaritmica).

5.2.2 Valutazione della mobilità dei droni

Percorso in città rettilineo

I primi test sulla mobilità dei droni sono stati fatti nella mappa cittadina (figura 5.1) utilizzando il percorso dritto, il più semplice dei due. Questi test sono gli stessi da cui si è ricavata la velocità di simulazione del sistema vista precedentemente.
Come si vede nella figura 5.1, i droni partono in un piazzale, un ambiente privo di ostacoli, per poi muoversi verso una delle vie principali della città. Con l'allontanarsi del drone leader si osserva come i droni, a causa del test dell'angolo acuto, si separano da alcuni loro vicini e cominciano a mettersi in fila dietro al leader (figura 5.6, **i dischi blu rappresentano i sensori, e al loro centro è presente il drone**).

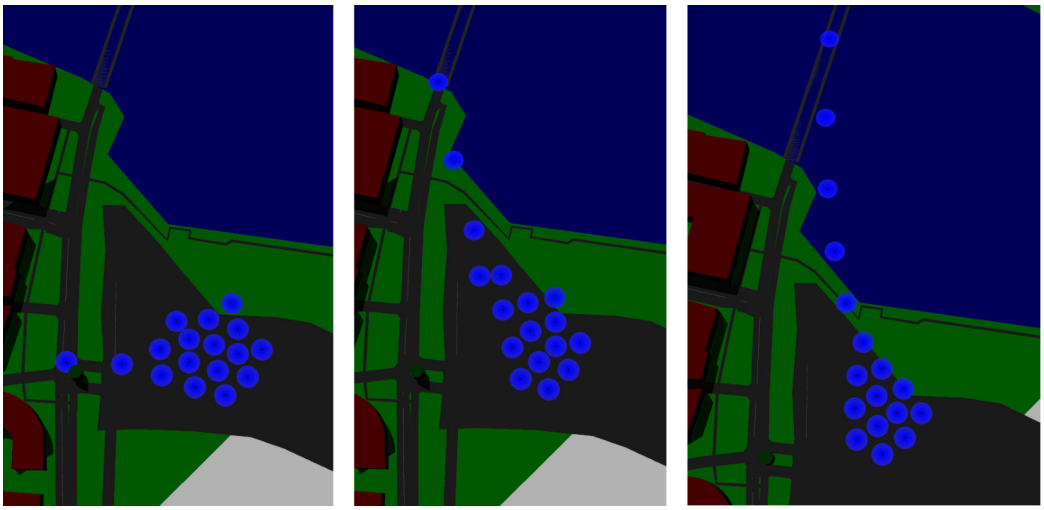


Figura 5.6: Il gruppo di droni comincia a formare una fila dietro al leader.

Una volta arrivati oltre il ponte, per i droni iniziano le difficoltà, in quanto si trovano per la prima volta degli ostacoli in mezzo al loro percorso (figura 5.7). Il sensore comincia a respingerli, ma la molla che hanno con il drone davanti a loro li spinge ad aggirare l'ostacolo. È però importante anche la forma dell'ostacolo, perchè i sensori del drone potrebbero spingere il drone nella direzione sbagliata.

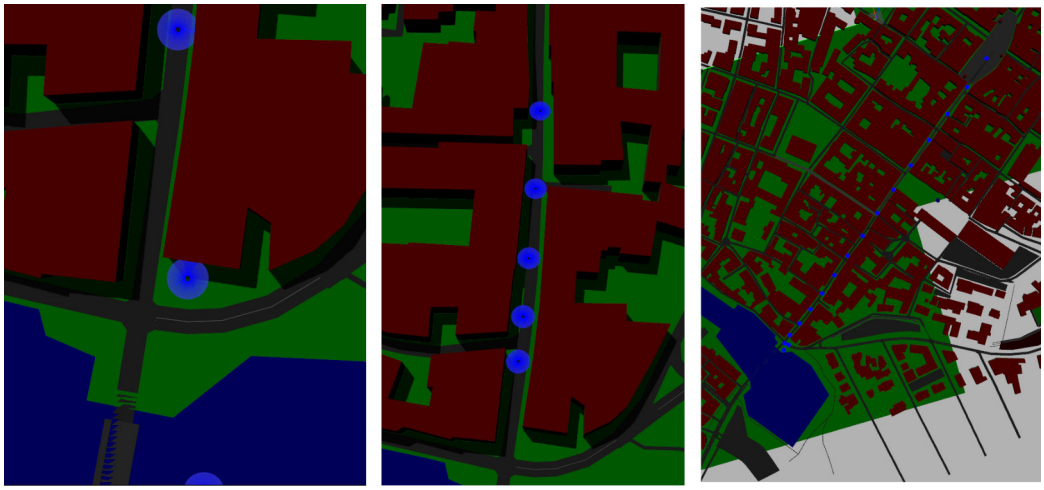


Figura 5.7: I droni incontrano un ostacolo all'ingresso della via, ma lo superano tutti.

Dopo aver condotto i test utilizzando i parametri base, si è provato ad aumentare la velocità del leader, portandola da 3 a 5 m/s. Anche qui i droni sono riusciti a seguire il leader, ma, oltre a viaggiare più velocemente, hanno avuto più problemi con l'ostacolo all'imbocco della via, in quanto hanno avuto meno tempo per superarlo e alcuni droni si sono avvicinati ad esso pericolosamente.

Dopo l'aumento di velocità si è provato a variare il targetLinkBudget, calandolo da 30 a 20 dBm. A seguito di questo, le molle dei droni li spingono ad essere più distanti tra di loro, e questo ha provocato 2 cose:

- nella simulazione con 2 droni, i due droni dello sciame non si sono messi in fila ma hanno proceduto affiancati, e questo ha creato problemi nell'imboccare la via (figura 5.8)
- nella simulazione con 16 droni, un drone della coda ha imboccato un vicolo cieco (figura 5.9)

Avendo visto la situazione di blocco con 16 droni, si è mandata avanti la simulazione oltre il tempo di 300 secondi e il leader più in là rispetto al punto di arrivo, per verificare l'avvenimento della perdita di un follower... quello

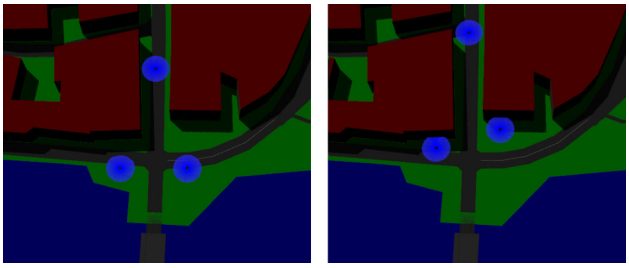


Figura 5.8: Coda di 2 droni affiancati presenta problemi a imboccare la via.

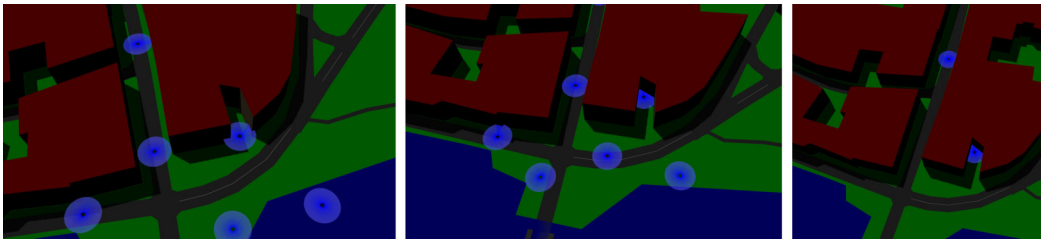


Figura 5.9: Un drone rimane bloccato in un vicolo.

che però si è verificato è un drone che ha attraversato un ostacolo per stare dietro al leader! *Le molle dei sensori non sono bastate ad evitare il contatto con l'ostacolo.*

Per evitare questo problema indesiderato si è diminuito il parametro k delle molle dello sciame. Si è visto che:

- con k = 2, il leader si distacca dagli altri droni prima ancora che il drone vada ad impigliarsi nel vicolo cieco (figura 5.10). Questo è dovuto ai droni dello sciame che attendono i compagni che hanno difficoltà a entrare nella via.

- con k = 5, il gruppo si distacca a causa del drone impigliato (vedi figura 5.9) ma stavolta il leader si porta dietro due altri droni (figura 5.11).

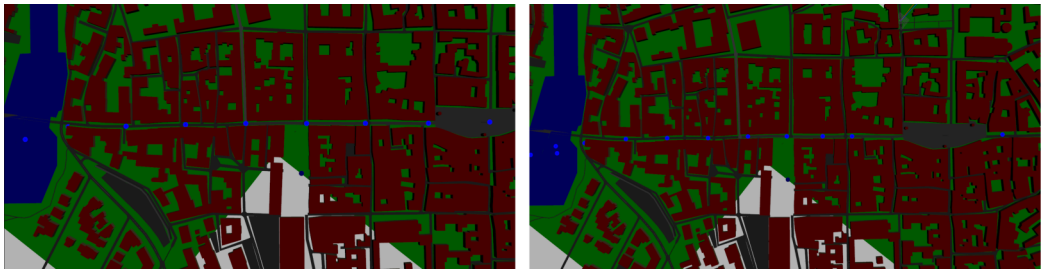


Figura 5.10: Distacco del leader con targetLinkBudget 20 dBm e parametro k = 2 (molle sciame).

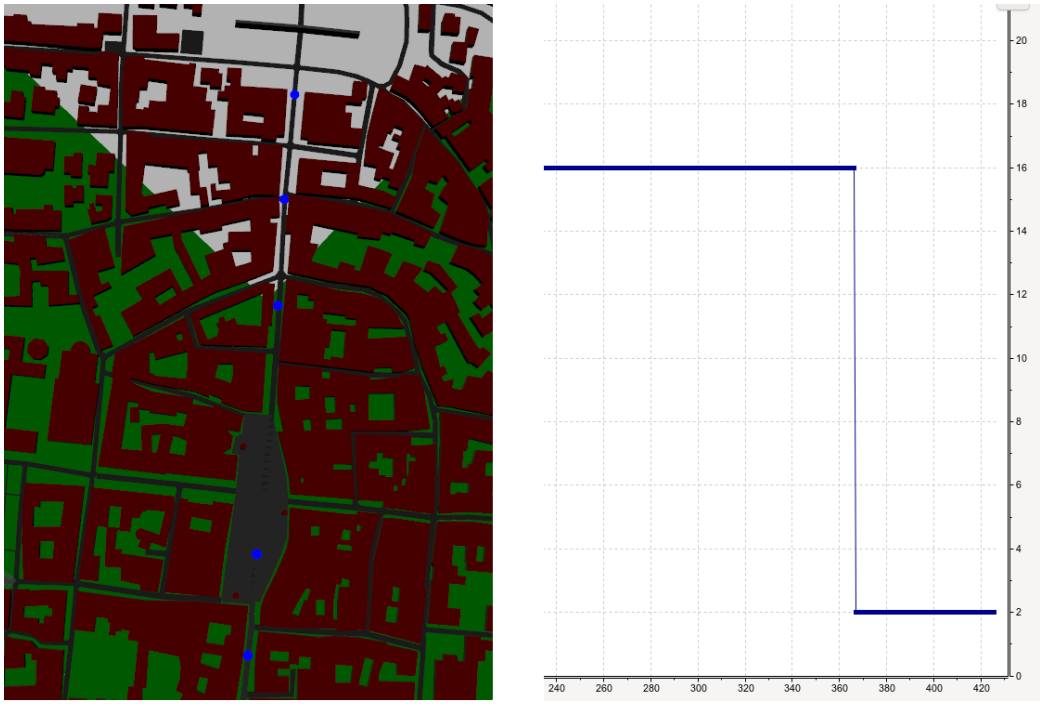


Figura 5.11: Distacco con targetLinkBudget 20 dBm e parametro k = 5 (molle sciame). Come si vede anche nel grafico di OMNeT++, i follower del leader calano da 16 a 2.

Avendo la simulazione con targetLinkBudget a 20 dBm avuto questi problemi, si è esaminato il grafico del LB medio soltanto per le altre due serie di simulazioni (figure 5.12 e 5.13). Guardando il grafico della prima simulazione (figura 5.12, simulazione con parametri base) si vedono delle differenze

variando il numero dei droni:

- con pochi droni il LB medio cala velocemente e improvvisamente, ma rimane poi stabile a un certo livello, mentre con l'aumentare dei droni cala in maniera più graduale.
- il livello di LB medio minimo raggiunto sembra diminuire con l'aumentare dei droni, ma con 8 e 16 droni si raggiunge un valore inferiore rispetto a con 20.
- con l'aumentare dei droni ci vuole più tempo a ritornare ad avere il LB medio pari al targetLinkBudget.

La causa dei fenomeni potrebbe essere principalmente la coda dello sciame che rimane ammassata per vario tempo, finchè non è costretta ad entrare in uno spazio stretto (ma le simulazioni effettuate non vanno abbastanza avanti, nel tempo e nello spazio, per far entrare tutti i droni per il gruppo da 16 e 20. È un caso da verificare in futuro).

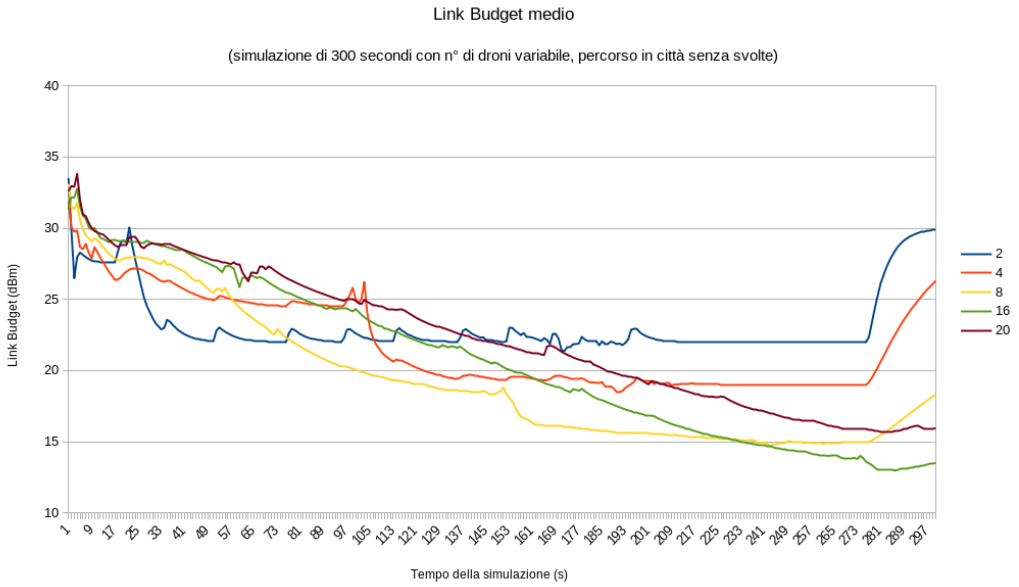


Figura 5.12: Il LB medio nel tempo nelle simulazioni in città (senza svolte) al variare del numero di droni.

Guardando il grafico per le simulazioni con la velocità del leader aumentata (figura 5.13) si vede un comportamento simile al precedente, in tutti e tre i punti. Si differenzia però per il livello di LB finale (i droni viaggiano più velocemente e ritornano prima al targetLinkBudget) e il livello di LB medio nel loro periodo di movimento (più basso rispetto a prima, in particolare per pochi droni).
C'è inoltre da notare come lo sciame di 8 droni raggiunga un LB medio molto basso, e la lentezza dello sciame da 20 droni a ritornare al targetLinkBudget.

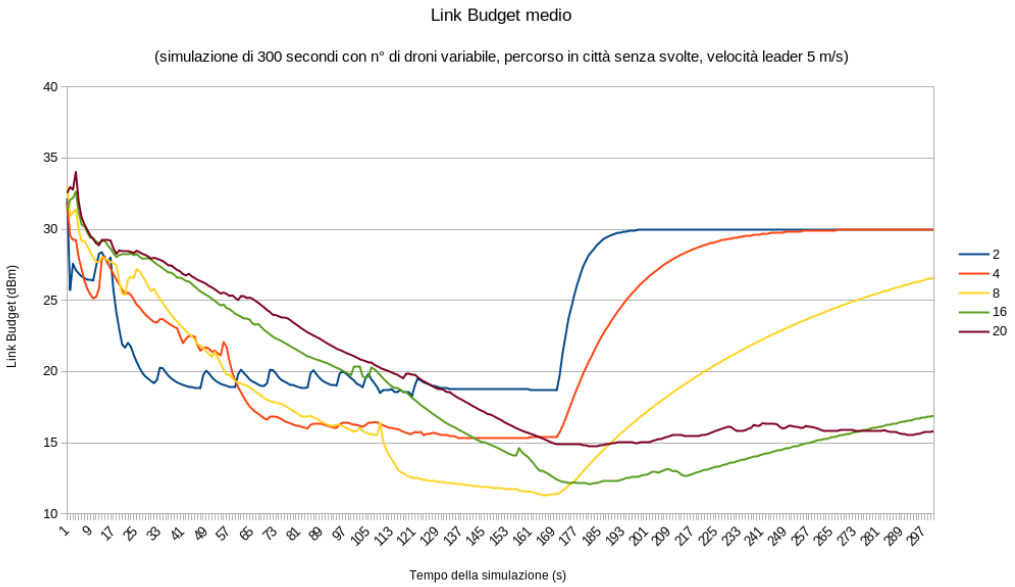


Figura 5.13: Il LB medio nel tempo nelle simulazioni in città (senza svolte), con velocità del leader aumentata a 5 m/s, al variare del numero di droni.

Percorso in città con svolte

Dopo aver effettuato i test con il percorso semplice (rettilineo) si è provato a far viaggiare i droni lungo il percorso con svolte. Si è provato inizialmente con 2 soli droni nello sciame, e il test ha dato esito positivo, con i droni che sono riusciti ad arrivare in fondo nonostante qualche difficoltà in certi punti. Si è quindi aumentato il numero di droni a 4, ma questo test è fallito alla prima

curva: incapaci di seguire il leader lungo il percorso, i droni hanno provato a prendere un'altra strada ma poi si sono schiantati contro un edificio. È stato testato anche il caso con 3 droni, e anche qui i droni hanno fallito alla prima curva, con il drone subito dietro al leader che ha effettuato la svolta con difficoltà (rischiando di scontrarsi contro l'edificio) ma che si è bloccato subito dopo.
Si è inizialmente pensato che il problema fosse causato dal particolare tipo di curva (vedi figura 5.14), con angolo maggiore di 90°, e dalla coda dello sciame che sembra rallenti i droni davanti.

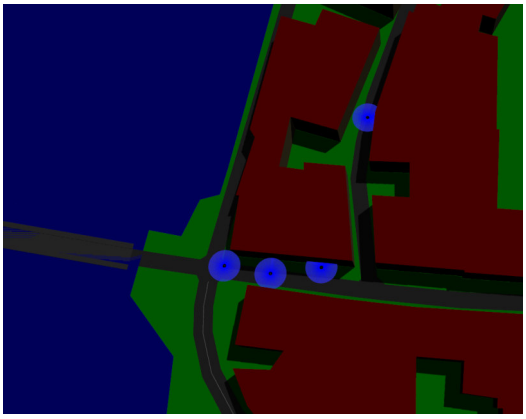


Figura 5.14: La prima curva del percorso in città con svolte.

Si è provato a ripetere il test con 2 droni anche per un targetLinkBudget di 20 dBm, ma i droni sono sono riusciti a effettuare la curva a causa della maggiore distanza dovuta alla molla con lunghezza a riposo maggiore. Dati i problemi a effettuare questa curva, e che i droni possono avere problemi a imboccare in generale certe vie, si è pensato che dovrebbe essere compito del leader scegliere una via più facilmente percorribile dal gruppo che lo segue. Si è quindi fatto una deviazione nel percorso con svolte, evitando la curva problematica e passando per una via più agibile.
Riprovando a mandare i 3 droni con targetLinkBudget di 30 dBm nella nuova strada si è riusciti, con qualche rischio di collisione, a farli arrivare alla destinazione, ma aumentando il numero di droni a 4 il test è nuovamente fallito a causa di una collisione accaduta nella curva in figura 5.15.

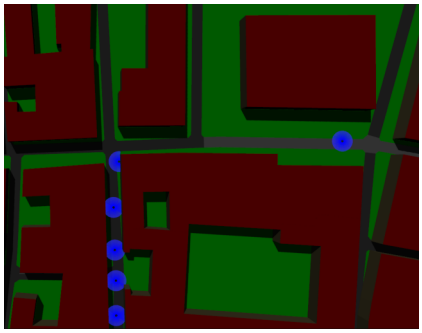


Figura 5.15: Il drone non riesce a superare questa curva e collide con il muro.

Si è quindi provato a diminuire la velocità del leader a 2 m/s. Sempre seguendo il percorso alternativo a quello base, si è fatto viaggiare lo sciame di 4 droni verso destinazione. Anche il gruppo di 5 droni riesce a fatica a completare il percorso, mentre quello di 6 fallisce, a causa di una collisione, sempre nella stessa curva di prima (figura 5.15). Il problemi sono il rallentamento dovuto alla coda, ma anche il fatto che non c'è nulla che spinge il drone a continuare lungo il percorso. A parte aumentare il parametro k delle molle, per far stare i droni più vicini, una soluzione potrebbe essere riconoscere che si sta andando contro un ostacolo come direzione ed esercitare una sorta di forza tangente come quella presente in [20] (o comunque una forza che lo spinga ad aggirare l'ostacolo).

Prima di provare nuovamente a ridurre la velocità del leader, si è voluto esaminare il comportamento dello sciame con la velocità del leader a 2 m/s lungo il percorso originale, ovvero con la curva iniziale problematica. Si è visto che i droni riescono a superare la curva, o ad aggirarla passando dalla via del percorso alternativo.
Si è quindi esaminato il comportamento del gruppo di droni con velocità del leader ridotta a 1 m/s. Nonostante qualche difficoltà il gruppo è riuscito a superare quasi tutte le curve, ma si è bloccato nella strettoia in figura 5.16.

Dopo questi test, si è visto come il sistema creato abbia difficoltà a superare anche semplici curve di 90°, e come non sia adatto a percorsi anche semplici ma sia limitato all'evitamento degli ostacoli nella sua mobilità.

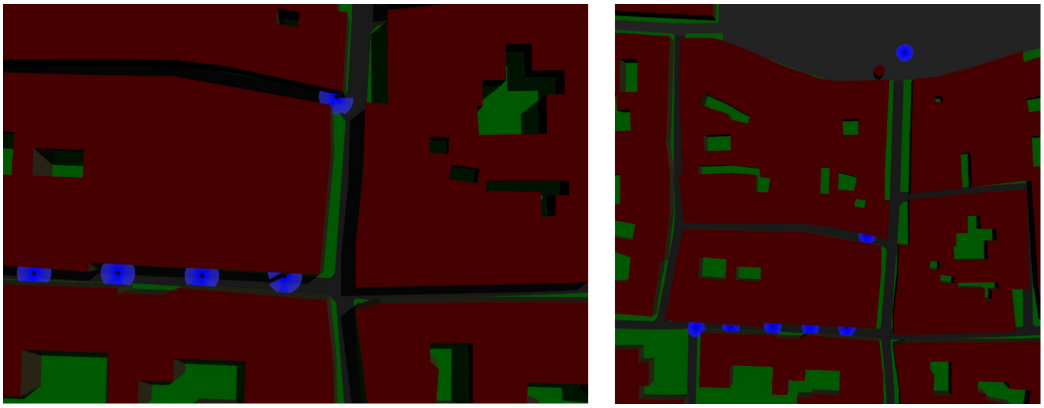


Figura 5.16: Andando a velocità molto ridotta (velocità del leader 1 m/s) i sensori conducono il drone in una via laterale.

Percorso in spazio aperto

Dopo il percorso in città si è andato a esaminare il LB medio nel caso di uno spazio aperto. È stato mantenuto il percorso di città rettilineo per confrontare i dati; si sono quindi effettuati i test al variare del numero dei droni, prima con parametri base, poi con l'aumento di velocità del leader a 5 m/s e infine con la riduzione del targetLinkBudget a 20 dBm.

Guardando i risultati dei test con i parametri base (figura 5.17) e confrontandoli con quelli del test in città (figura 5.12), la maggiore differenza sta nel caso con 4 droni, che in città cala improvvisamente per poi rimanere stabile a un livello di circa 19 dBm mentre in spazio aperto evita di calare e rimane stabile a circa 25 dBm. Anche nel caso di 8 droni non è presente un calo improvviso di LB medio e il risultato è migliore. Si pensa i cali siano dovuti all'ingresso nella via della città, e alla perdita della formazione di una sorta di coda che si mantiene in assenza di ostacoli (figura 5.18).
Per i casi restanti c'è poca differenza con i test in città; un comportamento diverso si vede alla fine, in particolare per il caso con 2 droni, per cui non si sa dare spiegazione della causa di una accellerazione improvvisa del drone di coda.

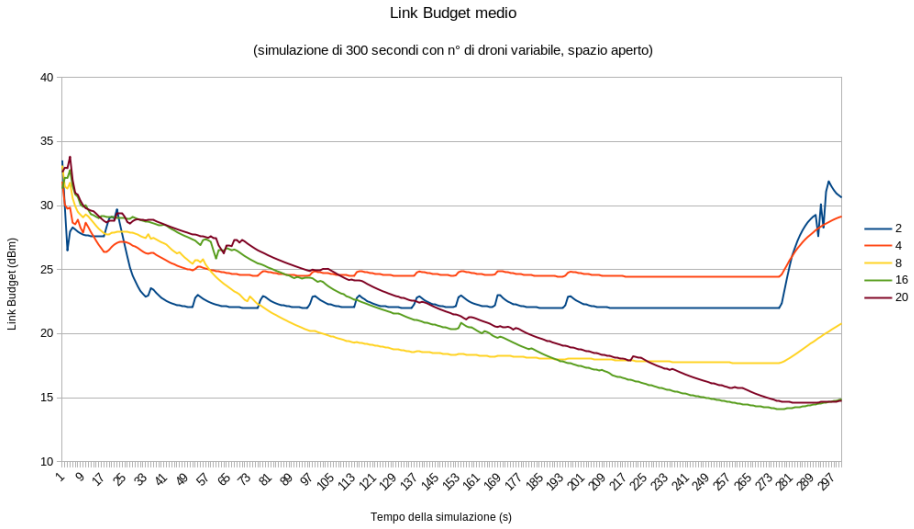


Figura 5.17: Il LB medio nel tempo nelle simulazioni in spazio aperto al variare del numero di droni.

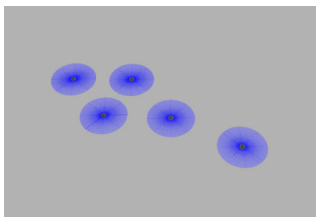


Figura 5.18: Coda nello sciame di 4 droni in spazio aperto con i parametri base.

Confrontando invece i risultati di città e spazio aperto relativi all'aumento di velocità del leader (figure 5.13 e 5.19), per i casi di 2, 4 e 16 droni ci sono poche differenze. Continua però a esserci il comportamento insolito verso la fine per il caso di 2 droni, e stavolta anche con 4.
Nel caso di 8 droni si vede come un calo improvviso verso metà simulazione sia sempre presente, anche se avviene poco più avanti nel tempo. Nel caso di 20 droni, invece, si nota un graduale aumento di LB verso la fine mentre nel caso in città il valore non mostra un chiaro andamento.

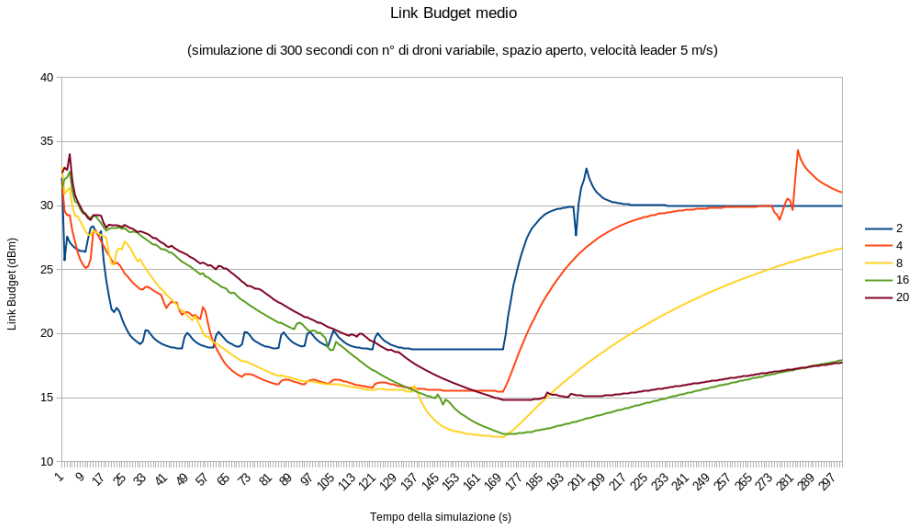


Figura 5.19: Il LB medio nel tempo nelle simulazioni in spazio aperto, con velocità del leader aumentata a 5 m/s, al variare del numero di droni.

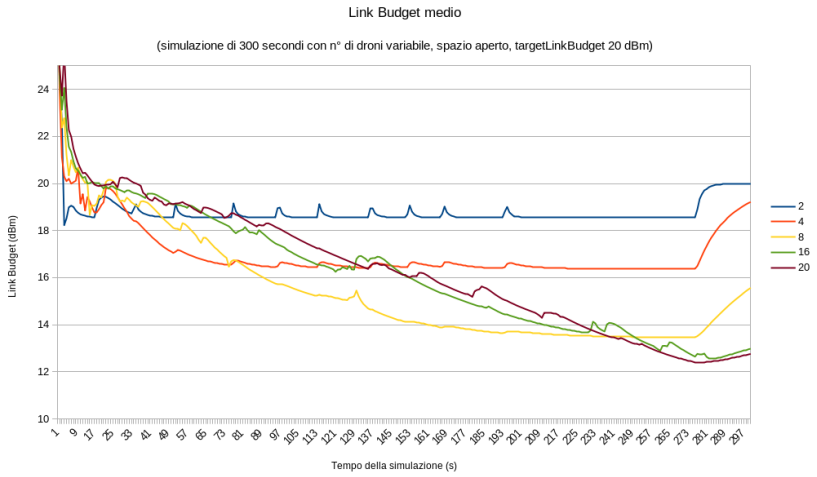


Figura 5.20: Il LB medio nel tempo nelle simulazioni in spazio aperto, con targetLinkBudget diminuito a 20 dBm, al variare del numero di droni.

Nel caso del targetLinkBudget a 20 dBm (figura 5.21) non si confrontano i valori con il test in città, in quanto in quest'ultimo i droni avevano problemi di mobilità e non sono stati registrati i valori. Per due droni la differenza

con il targetLinkBudget è minima, in quanto il valore rimane stabile a poco meno di 19 dBm, che è lo stesso valore del caso con la velocità aumentata mostrato sopra. I casi di 4 e 8 droni mostrano una chiara differenza, con il valore di LB medio che si fa più basso, mentre i casi di 16 e 20 droni sono molto simili tra loro.
Un comportamento che si è verificato in questo test è stato il mantenersi di una certa formazione nella coda del gruppo (figura 5.21), mentre invece si è visto che con l'aumento di velocità si formava meglio la fila dietro al leader. Sembra quindi che minore è il targetLinkBudget e la velocità del leader, maggiore è la presenza di una rete di droni nella coda.

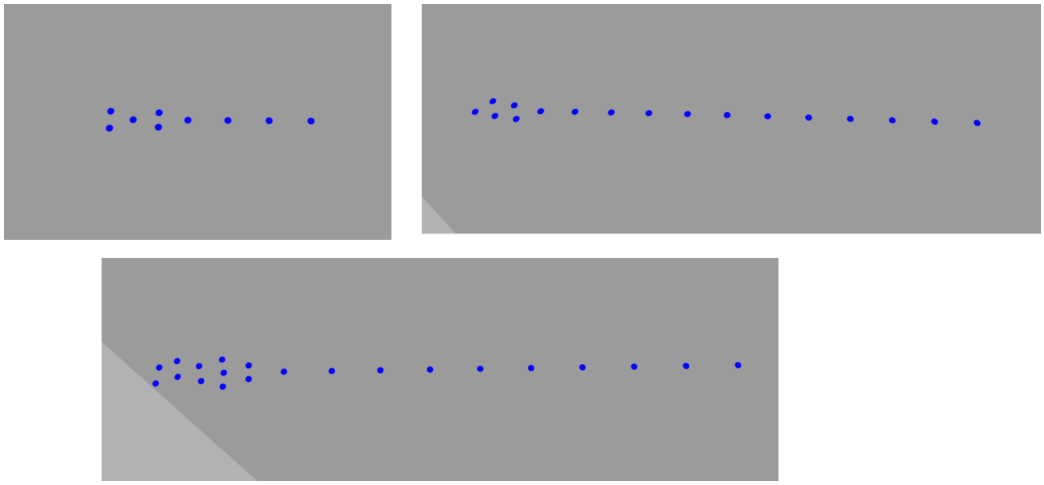


Figura 5.21: Le code dello sciame a fine percorso per 8, 16 e 20 droni (targetLinkBudget 20 dBm).

Nello scenario in campo aperto non ci sono ostacoli che possano provocare una separazione dello sciame in più gruppi, l'unico ostacolo potrebbe essere nel comportamento delle molle a determinati parametri, che però provocherebbe gli stessi problemi in altri scenari.
Nei tre casi visti (parametri base, velocità aumentata a 5 m/s, targetLinkBudget a 20 dBm) e con un numero di droni che va da 2 a 20 non si verifica però tale possibile problema.

Percorso in mezzo a ostacoli

Come ultimo scenario si è fatto procedere il gruppo di droni in mezzo a un'area contenente piccoli ostacoli, sparsi in modo da ostacolare il passaggio dei droni ma da non chiuderli in un vicolo cieco (è presente dello spazio tra di loro attraverso il quale il drone è in grado di passare).

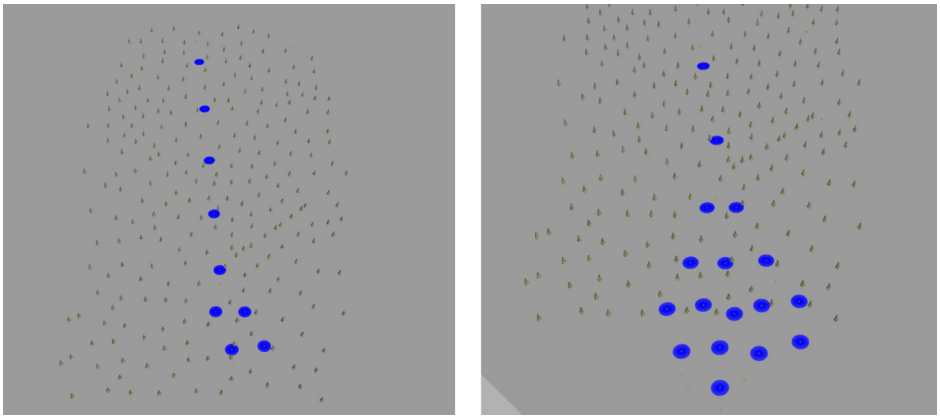


Figura 5.22: I droni seguono il leader evitando gli ostacoli.

Si è condotto il test come nelle modalità precedenti, e cioè variando il numero di droni e esaminando prima il comportamento dei droni con i parametri base, quindi con l'aumento di velocità del leader a 5 m/s e infine con la diminuzione del targetLinkBudget a 20 dBm. Si è deciso di vedere quanto i droni si avvicinano agli ostacoli al variare dei parametri, e per questo si è preso nota della minima distanza raggiunta da un drone qualsiasi dello sciame da un ostacolo, che può essere uno di quello dello scenario (gli alberi) o anche un altro drone. I risultati sono mostrati nella figura 5.23.

La distanza di un albero da un altro è variabile, e quindi i risultati sono da considerare indicativi. Dai risultati sembra che la velocità del leader aumentata abbia un effetto negativo, ovvero aumenti il rischio di collisioni. La distanza dall'ostacolo nel caso di 2 e 8 droni è molto più bassa, ma per 20 droni è in linea con gli altri due test.
Il test con il targetLinkBudget ridotto sembrava dare buoni risultati con pochi droni, ma con 16 e 20 dà i risultati peggiori; questo però potrebbe essere

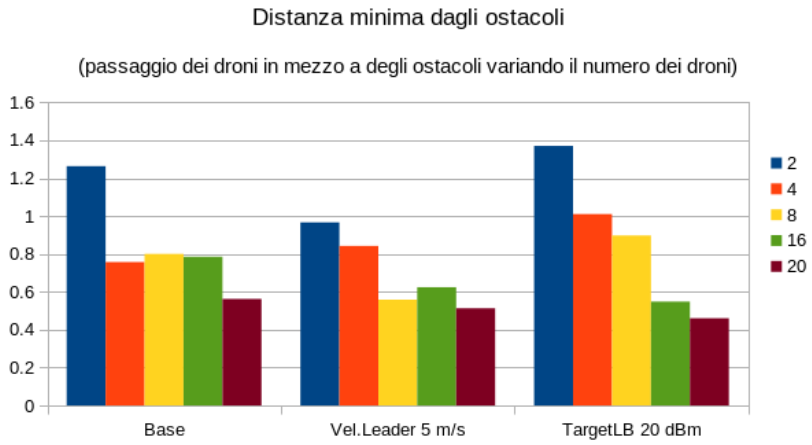


Figura 5.23: La distanza minima dagli ostacoli al variare del numero dei droni registrata nei vari test.

causa dello scenario, in quanto in questo test i droni tendono a stare di più in gruppo e potrebbero aver preso un percorso diverso dagli altri test.
Il test con i parametri base presenta una distanza maggiore per 16 droni, e fra 4, 8 e 16 non mostra quasi differenza, mentre il test con il targetLink-Budget diminuito è l'unico che mostra una diminuzione della distanza a ogni aumentare del numero di droni.

La presenza di ostacoli nel percorso dei droni potrebbe in teoria portare alla separazione dello sciame in più gruppi, in quanto i droni potrebbero subire una forza contraria e più forte rispetto a quella che li spinge ad andare verso il leader e che li faccia rimanere indietro, per esempio a causa di un ostacolo o a causa di un precedente drone a cui si è collegati che sta venendo trattenuto da un ostacolo. Nessuna separazione nè collisione è però avvenuta, e il gruppo è riuscito ad attraversare la foresta mantenendosi unito.

Valutazione finale della mobilità basata su molle

Avendo visto il comportamento delle molle nei vari scenari viene qui fatta una considerazione finale sulla validità del sistema delle molle.
La capacità del sistema di condurre i droni attraverso un percorso è molto

scarsa in quanto, come si è visto nello scenario di città, c'è una limitata capacità di aggirare l'ostacolo che richiede una velocità molto bassa e, anche in questo caso, può fallire. Il problema di questa limitata capacità consiste nell'assenza di una forza con tale obiettivo, che porti il drone ad aggirare un ostacolo, oltre che a evitarne il contatto.
Come si è visto nello scenario con ostacoli sparsi, però, l'algoritmo di evitamento ostacoli basato sulle molle dei sensori sembra sufficiente per evitare piccoli ostacoli. Tuttavia, se in questo scenario non ci sono state collisioni, queste si sono verificate nello scenario di città, facendo notare che la forza delle molle degli ostacoli non è sufficiente a contrastare la forza delle molle tra droni.
Una soluzione a questo problema è aumentare il parametro k della formula della molle per gli ostacoli, ma questo ha anche il difetto di rendere più difficile al drone attraversare delle strettoie. Inoltre, modificare il parametro k rende più difficile la presenza di una collisione, ma con le formule utilizzate è difficile assicurare che queste non si verificheranno.
Le formule utilizzate ricavano infatti la forza della molla da una divisione, dove a denominatore è presente un valore che tende sempre di più verso lo 0 (il Link Budget per le molle tra droni e la distanza dall'ostacolo per le molle per gli ostacoli), facendo crescere la forza sempre più velocemente al diminuire di questo valore: questo, nel caso di un drone che non riesce ad aggirare un ostacolo, porta a un continuo susseguirsi di spostamenti contrari e improvvisi nel drone, che lo rendono molto instabile nel movimento.
Sembrerebbe quindi una soluzione migliore, anche per una questione di sicurezza, mettere un controllo nel movimento del drone che sappia riconoscere che la forza delle molle lo sta portando ad avere una collisione e che eviti di applicare tale forza al drone.
Queste osservazioni sul comportamento delle molle sono state possibili grazie a Gazebo: a parte per l'uso dei sensori e la creazione di un ambiente realistico, l'ambiente 3D permette di osservare meglio gli sviluppi di una simulazione e, come si è visto nelle immagini, permette di visualizzare graficamente la

portata e lo stato dei sensori di prossimità.

Conclusioni

Sistemi di robot a mobilità autonoma dove i singoli robot comunicano fra loro per raggiungere un determinato obiettivo o per svolgere meglio le loro funzioni sono un argomento di crescente interesse, ma strumenti per la progettazione e analisi di questi sistemi, che riproducano accuratamente in una simulazione sia movimento e interazione del robot con l'ambiente che la gestione della comunicazione, sono ancora in fase di sviluppo.
In questa tesi si è cercato di creare un ambiente integrato dove effettuare simulazione di questo tipo di sistemi multi-agente, in particolare di un gruppo di droni con sensori di prossimità per evitare di provocare collisioni. I due simulatori si scambiano informazioni e partecipano, a turno, a far procedere la simulazione; OMNeT++ gestisce l'invio dei messaggi e ricava le nuove posizioni dei droni in base alla logica di mobilità da loro adottata, Gazebo riproduce i modelli ambientali della simulazione e ricava i dati dei sensori.
Dopo l'integrazione dei due simulatori, ottenuta facendo uso del meccanismo di comunicazione di Gazebo basato sui topic, si è riprodotto nel sistema integrato un caso d'uso, dove alcuni droni, organizzati tra loro tramite un sistema di molle virtuali, si spostano in un ambiente che può presentare ostacoli trascinati dal movimento un drone leader.

Si è quindi andati a valutare due sistemi: l'ambiente integrato (OMNeT++ & Gazebo) e la mobilità dello sciame di droni.

Nella valutazione del sistema integrato si è osservato le performance di Gazebo e del sistema integrato al variare di numeri di sensori e numero di droni. Si è notato che, per pochi droni/sensori, Gazebo introduce un grosso

ritardo nella simulazione dovuto al tempo di attesa dei dati dei sensori, legato al fatto che il dato del sensore viene pubblicato in maniera non bloccante sul topic, e quindi non viene inviato subito appena diventa disponibile. Se però, nel caso di un sistema con pochi droni/sensori, il ritardo introdotto nella simulazione è molto alto (simulazione 67 volte più lunga nel caso di 4 droni con 8 sensori, vedi figura 5.5), con l'aumentare del numero di droni il peso sul tempo totale di simulazione si abbassa (simulazione 12 volte più lunga nel caso di 20 droni) in quanto il carico di lavoro di OMNeT++ aumenta molto di più rispetto a quello di Gazebo (tempo di Gazebo in figura 5.3).

Per quanto riguarda la mobilità dei droni basata sulle molle (descritta nel Capitolo 4) la valutazione del sistema è stata effettuata in tre diversi scenari: spazio aperto, ambiente con ostacoli sparsi, ambiente di città realistico ottenuto con OpenStreetMap. Dai test si è visto come i droni tendano a formare una fila, soprattutto con valori di velocità o con targetLinkBudget alti, ma anche di come le loro possibilità di navigazione in un ambiente cittadino sia molto limitata: anche curve comuni ad angolo retto sono difficili da effettuare e, se la loro capacità di evitare gli ostacoli è migliore a velocità basse e targetLinkBudget alto (minore distanza tra un drone e l'altro), non si è però riusciti a completare il percorso con curve con velocità del leader a 1 m/s e targetLinkBudget a 30 dBm.
Inoltre, è capitata una situazione in cui il blocco di un drone in un vicolo cieco ha provocato la perdita di molti followers del leader, che avrebbero però avuto la possibilità di seguirlo. E anche con targetLinkBudget a 20 dBm si è provocato un distacco dove il leader si è separato dal resto del gruppo, rendendo il suo obiettivo di portare i droni a una certo luogo destinazione un completo fallimento.
I sensori sono però adatti a un ambiente con ostacoli sparsi, che i droni sono riusciti ad attraversare tenendo almeno 40 cm di distanza di sicurezza (figura 5.23).

Per quanto riguarda il LB medio dello sciame (calcolato sulle connessioni delle molle), si è visto che all'aumentare del numero dei droni nello sciame

cala in maniera più graduale ma scende di piè e ci mette più tempo a ritornare al targetLinkBudget. Il LB medio è legato alla distanza tra un drone e l'altro, e l'aumentare della velocità di viaggio lo fa diminuire; questo calo viene recuperato velocemente nel caso di pochi droni, ma nel caso di molti droni rimane per tempi molto più lunghi.
Inoltre, si è visto come il LB medio per drone sia più alto in fondo alla coda, dato che le distanze tra drone e drone sono maggiori in testa e minori in coda. Il calo graduale del LB medio è anche dovuto alla presenza di un gruppo di droni in coda, che si scompone lentamente nel tempo ma riesce in parte a mantenersi nel tempo minore è la velocità del leader e il targetLinkBudget (figura 5.21).
Un caso particolare è stato il test in spazio aperto con sciame di 4 droni e parametri base, che ha portato a un LB medio superiore al caso con 2 soli droni nello sciame (figura 5.17). Il motivo è stata la formazione adottata dai droni, due dei quali si sono allineati (figura 5.18). Questo fa venire in mente una possibile organizzazione dei droni alternativa a quella attuale, dove i droni cercano di organizzarsi a coppie invece che in fila singola, oppure a coppie alternate da singolo. Mantenendo per ogni drone un numero di molle in avanti verso il leader pari al numero di molle indietro verso la coda, la forza percepita da ogni drone sarebbe la stessa di come essere in fila uno a uno, ma la fila sarebbe più corta, le distanze tra i droni minori e quindi il LB medio più alto. Nel caso di spostamento in una zona priva di ostacoli (o con pochi semplici ostacoli) questa formazione di questo tipo potrebbe essere più indicata. Questa è però una considerazione del momento che richiede di essere verificata.

Le simulazioni effettuate e le osservazioni riportate sono state possibili grazie all'integrazione dei due sistemi, che permettono di unire gestione della rete di comunicazione, utilizzo dei sensori e scenario di simulazione realistico. Inoltre, varie osservazioni sono state rese possibili dall'ambiente 3D di Gazebo, che permette di osservare meglio l'andamento della simulazione.

Sviluppi futuri

Ci sono diverse cose in progetto come possibili sviluppi futuri. Per quanto riguarda il sistema integrato, ci sono in mente funzionalità aggiuntive e alcune cose da sistemare:

- per rendere i test più veloci si è specificato nel plugin di Gazebo di eseguire solo un millisecondo di simulazione per ogni 0.1 s di tempo simulato (nei test il tempo di aggiornamento delle posizioni dei droni e dei sensori coincidono), ma questa dovrebbe essere una funzionalità che OMNeT++ specifica al plugin tramite messaggio di sincronizzazione

- il metodo utilizzato nel plugin di Gazebo per ottenere i dati dei sensori al momento funziona solo se l'intervallo di tempo simulato da Gazebo non porta a molteplici aggiornamenti dello stesso sensore. La gestione dei sensori in Gazebo va rivista

- al momento i sensori di un drone sono specificati all'interno del file .sdf del modello, ma sarebbe utile poter specificare i sensori direttamente in OMNeT++, e usare il file .sdf solo per indicare il modello da utilizzare

- per avere una simulazione più realistica un possibile sviluppo è quello di considerare in OMNeT++ l'influenza degli eventuali ostacoli ambientali nella comunicazione dei messaggi tra droni

- una funzionalità a cui si è interessati è quella dell'elaborazione in OMNeT++ di immagini ricavate da Gazebo, utilizzando algoritmi di computer vision. Questo richiede la gestione dei sensori camera di Gazebo e il trasferimento di tali immagini da Gazebo a OMNeT++

- per limiti di tempo i dati dei sensori vengono momentaneamente passati al modulo di mobilità da parte del GazeboSynchronizer, mentre dovrebbe essere il modulo di mobilità a prendere i dati dai sensori dai moduli dei sensori

Per quanto riguarda il sistema di mobilità dei droni, ciò che si ipotizza per il futuro è:

- mettere un controllo che eviti di mandare il drone contro un ostacolo quando la forza delle molle lo spingerebbe a farlo. Questo è importante per una questione di sicurezza
- prevedere un modo per far aggirare al drone un ostacolo. Si potrebbe riconoscere quando un drone è spinto ad andare verso a un ostacolo e trasformare quella forza in modo da aumentare lo spostamento nella direzione di viaggio
- effettuare più test. Per esempio, non si è provato a diminuire la portata dei sensori, la frequenza di invio dei messaggi broadcast, il parametro k dei sensori
- al momento il drone leader viaggia sempre a velocità costante, ma si potrebbe considerare di farlo rallentare quando i droni dietro di lui fanno fatica a seguirlo (per esempio aggiungendo delle molle come in [20])
- provare a modificare le formule delle molle, sia quella tra due droni che quella per gli ostacoli, che al momento generano entrambe molta forza con basso LB o con bassa distanza dall'ostacolo provocando nel drone cambi di velocità improvvisi
- rendere realistico il movimento del drone, che al momento non tiene conto della sua velocità attuale e permette cambi di velocità e direzione istantanei

Bibliografia

[1] A. Varga , Omnet++, in: Modeling and Tools for Network Simulation, Springer, 2010, pp. 35-59

[2] Open Source Robotics Foundation, Gazebo, http://gazebosim.org

[3] C. Sommer and F. Dressler, "Progressing toward realistic mobility models in VANET simulations", in *IEEE Communications Magazine*, vol. 46, no. 11, pp. 132-137, Nov 2008

[4] D. Krajzewicz, G. Hertkorn, C. Rossel, P. Wagner, "SUMO (Simulation of Urban MObility); An Open-Source Traffic Simulation", in *Proc. Fourth Middle East Symp. Simulation and Modelling (MESM '02)*, pp. 183-187, Sept. 2002

[5] M. Piòrkowski, M. Raya, A.L. Lugo, P. Papadimitratos, M. Grossglauser, J. Hubaux, "TraNS: Realistic Joint Traffic and Network Simulator for VANETs", in *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 12, Jan 2008

[6] Various Academic Subjects, Network Simulator (Version 2), available from http://nsnam.sourceforge.net/wiki/index.php/User_Information

[7] C. Sommer, R. German and F. Dressler, Â´Â´Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis", in *IEEE Transactions on Mobile Computing*, vol. 10, no. 1, pp. 3-15, Jan 2011

[8] V. Kumar, L. Lin, D. Krajzewicz, F. Hrizi, O. Martinez, J. Gozalvez, R. Bauza, Â´Â´iTETRIS: Adaptation of ITS Technologies for Large Scale Integrated Simulation", in *IEEE 71st Vehicular Technology Conference*, Taipei, pp. 1-5, July 2010

[9] Network Simulator (Version 3), available from https://www.nsnam.org/

[10] J. Xu, L. Xie, T.G. Toh, Y.K. Toh, "HNMSim: A 3D multi-purpose hybrid networked multi-agent simulator", in *Proceedings of the 31st Chinese Control Conference*, 2012

[11] M. Kudelski, L.M. Gambardella, G.A. Di Caro, "RoboNetSim: An Integrated Framework for Multi-robot and Network Simulation", in *Robotics and Autonomous Systems*, Vol. 61, pp. 483-496, May 2013

[12] N.R. Zema, A. Trotta, E. Natalizio, M. Di Felice, L. Bononi, "The CUSCUS simulator for distributed networked control systems: Architecture and use-cases", in *Ad Hoc Networks*, Vol. 68, pp. 33-47, Jan 2018

[13] Heudiasyc Laboratory, UMR CNRS 7253, FL-AIR: Framework Libre air, available from https://uav.hds.utc.fr/software-flair/

[14] B. Vieira, R. Severino, E.V. Filho, A. Koubaa, E. Tovar "COPADRIVe - A Realistic Simulation Framework for Cooperative Autonomous Driving Applications", in *IEEE International Conference on Connected Vehicles and Expo (ICCVE)*, 2019

[15] A. Trotta, M. Di Felice, L. Bedogni, L. Bononi, F. Panzieri, "Connectivity recovery in post-disaster scenarios through Cognitive Radio swarms", in *Computer Networks*, Vol. 91, pp. 68-89, Nov 2015

[16] K. Derr, M. Manic, "Extended virtual spring mesh (EVSM): the distributed self-organizing mobile ad hoc network for area exploration", in *IEEE Transactions on Industrial Electronics*, Vol. 58, pp. 5424-5437, Dec 2011

[17] B. Shucker, J.K. Bennett, "Virtual Spring Mesh Algorithms for Control of Distributed Robotic Macrosensors", 2005

(available at https://www.semanticscholar.org/paper/Virtual-Spring-Mesh-Algorithms-for-Control-of-%3B-Shucker-Bennett/efc9ee71561a712fa88eefaaa33ae316fc035e50)

[18] N. Gallardo, K. Pai, B.A. Erol, P. Benavidez, M. Jamshidi, "Formation control implementation using kobuki turtlebots and parrot bebop drone", in *2016 World Automation Congress (WAC)*, pp. 1-6, July 2016

[19] K. Piemngam, I. Nilkhamhang, P. Bunnun, "A Virtual Spring Damper Method for Formation Control of the Multi Omni-directional Robots in Cooperative Transportation", in *11th International Conference on Information Technology and Electrical Engineering (ICITEE)*, Dec 2019

[20] P. Urcola, L. Riazuelo, M. Lazaro, L. Montano, "Cooperative navigation using environment compliant robot formations", pp. 2789-2794, Oct 2008

[21] A. Howard, M.J. Mataric, and G.S. Sukhatme, "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem", in *6th International Symposium on Distributed Autonomous Robotic Systems (DARS)*, June 2002

[22] W.M. Spears, D.F. Gordon, "Using artificial physics to control agents", in *Proceedings of IEEE International Conference on Information, Intelligence, and Systems*, 1999

[23] D.F. Gordon, W.M. Spears, O.Sokolsky, I. Lee, "Distributed spatial control, global monitoring and steering of mobile agents", in *Proceed ings of IEEE International Conference on Information, Intelligence, and Systems*, 1999

[24] D.F. Gordon-Spears, W.M. Spears, "Analysis of a phase transition in a physics-based multiagent system", in *Proceedings of NASA God-*

dard/IEEE Workshop on Formal Approaches to Agent-Based Systems, 2002

[25] OMNeT++ team, INET framework, https://inet.omnetpp.org/

[26] Google, Protocol Buffers, https://developers.google.com/protocol-buffers

[27] C.M. Kohlhoff, boost.ASIO, https://www.boost.org/doc/libs/1_72_0/doc/html/boost_asio.html

[28] Open Source Robotics Foundation, Ros, http://www.ros.org

[29] https://developer.parrot.com/docs/sphinx/openstreetmap.html