ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

Corso di Laurea in Ingegneria e Scienze Informatiche

Analisi dell'impiego di SOG e COG in algoritmi di clustering per reti marittime

Tesi di laurea in:
Programmazione ad Oggetti

Relatore
Prof. Danilo Pianini

Candidato
Lorenzo Antonioli

Correlatore
Dott. Martina Baiardi

I Sessione di Laurea
Anno Accademico 2025-2026

Sommario

Il settore marittimo sta subendo una rapida trasformazione con l'avvento delle navi a guida autonoma, che richiedono un'elevata larghezza di banda per condividere grandi flussi di dati. Attualmente, le reti di comunicazione tradizionali, come l'Automatic Identification System (AIS) su frequenze Very High Frequency (VHF), offrono data rate limitati e non sono in grado di supportare tali carichi. Per superare queste limitazioni, soluzioni recenti propongono algoritmi multi-hop auto-organizzanti basati su programmazione aggregata per raggruppare le navi in cluster (gruppi) cooperativi. Tuttavia, l'algoritmo di riferimento aggrega i nodi valutandone esclusivamente la posizione, ignorando la rotta e la velocità delle navi. Ciò causa una continua variazione della topologia di rete e instabilità nei collegamenti. L'obiettivo di questa tesi è implementare una riconfigurazione della rete consapevole della navigazione. Il lavoro estrae i parametri Speed Over Ground (SOG) (velocità) e Course Over Ground (COG) (rotta) dai payload dei messaggi AIS per calcolare una penalità di mobilità. Questa funzione disincentiva la formazione di collegamenti tra imbarcazioni con cinematica molto diversa. L'algoritmo è stato implementato nel framework Collektive e testato nel simulatore Alchemist. Le simulazioni dimostrano che l'introduzione dell'affinità navigazionale modifica la topologia di rete e che l'aggiunta di SOG e COG non ha migliorato il data rate complessivo. L'analisi dimostra che i dati di movimento delle imbarcazioni non sono sufficienti per migliorare sensibilmente l'efficienza comunicativa della rete e che ulteriori fattori dovranno essere considerati in future strategie di ottimizzazione.

A Vittoria, che ci ha creduto prima che ci credessi io, che mi ha spinto a diventare la versione migliore di me. Grazie per amarmi più di quanto io ami me stesso e per darmi la vita più bella che potessi desiderare. Questo traguardo è anche tuo.

Indice

Capitolo 1

Introduzione

Il settore marittimo rappresenta l'infrastruttura portante del commercio globale, ma si trova oggi ad affrontare una fase di profonda trasformazione tecnologica orientata all'efficienza, alla sicurezza e alla sostenibilità. In questo contesto, l'emergere delle Maritime Autonomous Surface Ships (MASS) promette di rivoluzionare la navigazione riducendo l'errore umano e ottimizzando i processi operativi [BPAFT25, AFBPT25]. Tuttavia, l'inadeguatezza delle reti di comunicazione marittima convenzionali frena il passaggio verso navi interamente o parzialmente autonome. Mentre i moderni dispositivi a bordo (come telecamere 4K, Light Detection and Ranging (LiDAR) e radar) generano massicci flussi di dati necessari per mantenere un'adeguata situational awareness, le comunicazioni tradizionali offrono una larghezza di banda limitata, insufficiente a supportare lo scambio di streaming video o di point clouds in tempo reale [BPAFT25, BPW14]. Per superare questo collo di bottiglia senza dipendere da costose connessioni satellitari, emerge la necessità di adottare paradigmi come Aggregate Programming, puntando a creare reti multi-hop e auto-organizzanti direttamente tra le imbarcazioni e le infrastrutture costiere [BPV15, CPVN19].

La motivazione alla base di questo lavoro di tesi nasce dall'analisi dei limiti delle attuali architetture di rete in ambito marittimo. Poiché in questo tipo di ambiente la connettività è sparsa e i data rate sono bassi, l'articolo *Robust Communication through Collective Adaptive Relay Schemes for Maritime Vessels* ha valutato approcci in cui i dati vengono raccolti localmente e poi riassunti in punti

strategici della rete [BPAFT25]. Idealmente, le imbarcazioni si auto-organizzano in gruppi (definiti *cluster*) all'interno dei quali i dati possono essere trasmessi in modo affidabile verso un nodo designato come sintetizzatore, ovvero il leader del cluster [BPAFT25]. Il leader ha il compito di comprimere i flussi raccolti, incluso il proprio, e inoltrare questo output sintetizzato alla stazione a terra o a un relay nel cluster successivo, sfruttando torri cellulari 5G [BPAFT25]. Sebbene questa architettura (identificata come la "Fase 1" dell'agenda *CoMPass*) migliori il data rate in scenari simulati, presenta una debolezza: la formazione dei cluster è basata solo su posizioni statiche (che non tengono conto della velocità o della rotta delle navi) [BPAFT25, AFBPT25]. L'uso di queste metriche non è infatti sufficiente, in quanto le imbarcazioni con traiettorie e velocità differenti dovrebbero essere scoraggiate dal formare collegamenti di comunicazione che diventerebbero rapidamente instabili [AFBPT25].

L'obiettivo principale di questo lavoro di tesi è implementare la "Fase 2" della roadmap CoMPass, ossia la riconfigurazione della rete consapevole della navigazione [AFBPT25]. Partendo dal framework di comunicazione stabilito, questa fase incorpora il posizionamento dinamico delle navi con l'intento di rendere più realistica e selettiva la topologia della rete. Nello specifico, questa modifica utilizza i dati di navigazione estraendo dal payload dei messaggi AIS i valori di SOG e COG per perfezionare gli algoritmi di clustering e le strategie di comunicazione. Valutando questi parametri, le navi che si muovono in direzioni e velocità simili ottengono la priorità per l'appartenenza al cluster mentre le imbarcazioni con traiettorie e velocità divergenti vengono scoraggiate dal formare collegamenti di comunicazione che diventerebbero rapidamente obsoleti.

Il documento è strutturato in capitoli secondo la seguente organizzazione. Il capitolo 2 espone i fondamenti teorici e le tecnologie di riferimento, descrivendo i sistemi di comunicazione marittima convenzionali (con particolare focus sulla struttura e i messaggi AIS), Aggregate Programming, Field Calculus e gli strumenti di simulazione utilizzati. Viene inoltre descritto in dettaglio l'esperimento precedente. Il capitolo 3 è dedicato all'analisi del problema. Vengono descritti: i limiti dell'algoritmo esistente, gli obiettivi della tesi e i requisiti funzionali e non funzionali richiesti per la realizzazione del nuovo sistema. Il capitolo 4 espone il design e l'implementazione, illustrando la metodologia di utilizzo e integrazione

dei nuovi dati all'interno del programma e le modifiche all'algoritmo. Il capitolo 5 presenta la comparazione dei risultati ottenuti con quelli precedenti, utilizzando le stesse metriche di valutazione. Infine si discute delle conclusioni e dei possibili sviluppi futuri del lavoro svolto.

Capitolo 2

Background

In questo capitolo viene presentato il quadro teorico e tecnologico della tesi. La sezione 2.1 analizza i sistemi di comunicazione marittima convenzionali (con focus sull'AIS), evidenziando la differenza tra la loro banda limitata e l'elevata quantità di dati generata dalle navi a guida autonoma (MASS) [BPAFT25, AFBPT25].

Per superare questo limite, la sezione 2.2 introduce l'Aggregate Programming e il pattern Self-organising Coordination Regions (SCR) (tramite il framework Collektive). In un ambiente marittimo, caratterizzato da un'elevata mobilità delle navi e connettività sparsa, questi approcci risultano efficaci poiché permettono di affrontare tali sfide creando protocolli di comunicazione resilienti e in grado di auto-organizzarsi dinamicamente [BPV15, CPVN19, BPAFT25].

Infine, la sezione 2.3 illustra il ruolo del Modeling & Simulation tramite il simulatore Alchemist. Poiché i test sul campo comportano limiti fisici ed elevati costi operativi, la simulazione offre l'ambiente virtuale ideale per sviluppare e valutare le prestazioni di tali reti cooperative [BPAFT25, PMV13].

2.1 Comunicazioni marittime e AIS

Le comunicazioni marittime si basano su una combinazione di sistemi radio VHF e satellitari (Satellite Communications (SatCom)) per garantire la sicurezza della navigazione e l'efficienza delle operazioni [BPAFT25]. Mentre i sistemi satellitari assicurano una copertura globale (comportando latenze e costi maggiori), i siste-

mi VHF supportano lo scambio di dati a corto raggio (in linea di vista o Line of Sight, tipicamente a 40 km a seconda del meteo) per i collegamenti sia tra imbarcazioni (Ship to Vessel (S2V)) sia verso le stazioni costiere (Ship to Shore (S2S)) [BPAFT25]. Una delle tecnologie fondamentali basate sui sistemi radio è l'AIS. L'AIS è un sistema di identificazione automatica che permette alle navi di trasmettere e ricevere in modo continuo informazioni sulla propria posizione, rotta, velocità e stato di navigazione, utilizzando la banda radio VHF dedicata [Int26].

2.1.1 Funzionamento del sistema AIS

L'AIS è nato come strumento per aumentare la sicurezza in mare, affiancando i radar per ridurre il rischio di collisioni e migliorare la "consapevolezza situazionale" dei comandanti [Int26, Uni15]. Ogni transponder AIS invia periodicamente messaggi brevi (tipicamente ogni 2–6 secondi) che includono sia dati statici (identificativo International Maritime Organization (IMO), Maritime Mobile Service Identity (MMSI), tipo di nave, dimensioni) sia dati dinamici (posizione Global Positioning System (GPS), rotta, velocità, stato di navigazione), oltre a dati di viaggio come la destinazione prevista [Int26]. Questi messaggi possono essere letti da altri transponder a bordo (modalità S2V) e da stazioni costiere (S2S), permettendo il monitoraggio del traffico da parte delle autorità (Vessel Traffic Service (VTS), Guardia Costiera) [Int26, Uni15] e anche da servizi commerciali (es. portali come MarineTraffic[1] o VesselFinder[2]). La portata tipica è di alcune decine di miglia nautiche, limitata dalla propagazione VHF e dalle condizioni meteorologiche [BPAFT25], ma può essere estesa tramite reti satellitari Satellite AIS (AIS-sat) [Int26].

2.1.2 Struttura e codifica del messaggio AIS

I messaggi AIS non vengono trasmessi in chiaro come testo, ma sono codificati in formato binario compresso (utilizzando una codifica a 6 bit) per minimizzare l'occupazione della banda radio VHF [Int26]. I diversi tipi di messaggi (da 1 a 27) rispondono a specifiche esigenze; tra questi, i messaggi di Tipo 1, 2 e 3 sono i più

[1] `https://www.marinetraffic.com`
[2] `https://www.vesselfinder.com`

frequenti e vengono impiegati dalle navi di Classe A per comunicare periodicamente i dati di posizione dinamica [Int26]. La tabella 2.1 descrive dettagliatamente la ripartizione dei bit all'interno di un frame AIS di Posizione standard.

Bit	**Nome Campo**	**Descrizione**
6	Message ID	Identifica il tipo di messaggio (valori: 1, 2 o 3).
2	Repeat Indicator	Numero di ripetizioni del messaggio.
30	Source ID (MMSI)	Codice univoco di 9 cifre della nave.
4	Navigational Status	Stato della nave (es. 0 = in navigazione, 1 = all'ancora, 5 = ormeggiata).
8	ROT	Velocità di accostata (virata) a destra o sinistra.
10	SOG	Velocità rispetto al fondo (risoluzione 0.1 nodi).
1	Position Accuracy	Accuratezza del GPS (1 = alta, ≤ 10 m; 0 = bassa, > 10 m).
28	Longitude	Longitudine in 1/10000 di minuto d'arco.
27	Latitude	Latitudine in 1/10000 di minuto d'arco.
12	COG	Rotta rispetto al fondo (in decimi di grado, 0–3599).
9	True Heading	Prua vera della nave (0–359 gradi).
6	Time Stamp	Secondo della coordinata temporale UTC di generazione (0–59).
2	Special Manoeuvre	Indicatore di manovre speciali.
2	Spare	Bit di riserva inutilizzati (impostati a 0).
1	Transmit Power	Potenza di trasmissione.
1	RAIM flag	Indicatore di integrità del ricevitore GPS.
19	Communication State	Dati di sincronizzazione e gestione degli slot temporali.
168	**Totale**	Struttura che occupa 1 slot di trasmissione.

Tabella 2.1: Struttura del payload di un messaggio AIS di Posizione (Tipo 1, 2, 3).

2.1.3 Aspetti normativi e applicazioni

L'uso del Sistema di Identificazione Automatica è regolato dall'IMO ed è obbligatorio ai sensi della convenzione Safety Of Life At Sea (SOLAS) per le navi commerciali in viaggio internazionale con stazza lorda pari o superiore a 300 Gross Tonnage (GT), e per tutte le navi passeggeri indipendentemente dalle loro dimensioni [BPW14, Uni15]. Oltre all'identificazione e al monitoraggio dei corridoi marittimi, l'AIS è oggi impiegato in diversi ambiti. Tra questi spicca la prevenzione delle collisioni in tempo reale (Closest Point of Approach (CPA)), che permette ai sistemi di bordo di innescare allarmi visivi e acustici qualora si rilevi una possibile collisione [BPW14]. Inoltre, il sistema risulta fondamentale per le operazioni di ricerca e soccorso (Search and Rescue (SAR)) e la gestione di uomini in mare (Man OverBoard (MOB)): i trasmettitori di soccorso (es. AIS-Search and Rescue Transponder (AIS-SART)) si attivano a contatto con l'acqua trasmettendo la posizione GPS a tutte le unità limitrofe, utilizzando messaggi specifici e un prefisso MMSI dedicato [BPW14, Int26]. Un altro campo in cui viene utilizzato il Sistema di Identificazione Automatica è la segnalazione di ostacoli tramite ausili alla navigazione (Aids to Navigation (AtoN)): boe, fari o stazioni possono trasmettere le proprie coordinate o simulare ostacoli virtuali (Virtual AtoN) per deviare il traffico da aree pericolose o fondali bassi [BPW14, Int26]. Infine, l'AIS agevola il monitoraggio ambientale per rintracciare le navi responsabili di versamenti illegali di petrolio in mare [BPW14].

2.1.4 Limiti e problemi aperti

Nonostante l'ampia diffusione, sono presenti alcuni limiti e vulnerabilità dell'AIS. In primo luogo, la sua natura non autenticata lo espone a vulnerabilità informatiche. I dati generati a bordo possono essere manipolati (*spoofed*) per creare navi inesistenti o falsi allarmi SAR/CPA, alterati sovrascrivendo i segnali (*hijacking*) oppure resi indisponibili tramite attacchi radio *Denial of Service*, come il blocco del salto di frequenza (*frequency hopping*) o l'esaurimento degli slot disponibili (*slot starvation*) [BPW14]. Dal punto di vista fisico, il canale della trasmissione VHF offre una larghezza di banda molto stretta (circa 10 kbps) a fronte di una copertura limitata Line of Sight (tipicamente 40 km, fortemente dipendente dalle

condizioni meteorologiche) [BPAFT25]. Di conseguenza, le attuali reti risultano incompatibili con i requisiti delle navi a guida autonoma (MASS): questi mezzi richiedono sensori avanzati come telecamere ad alta risoluzione, LiDAR e radar, generando flussi di dati continui nell'ordine dei Mbps che le attuali reti marittime non sono in grado di sostenere [BPAFT25].

In particolare, questo lavoro di tesi affronterà l'ultimo problema elencato sopra, utilizzando la programmazione aggregata (sezione 2.2).

2.2 Aggregate Programming, SCR e Collektive

2.2.1 Programmazione Aggregata (AP)

La programmazione di sistemi distribuiti moderni, come l'Internet of Things (IoT) o le reti navali, risulta complessa a causa dell'eterogeneità dei dispositivi, dei guasti frequenti e della necessità di interazioni basate sulla prossimità fisica [BPV15]. Per far fronte a questi problemi, Aggregate Programming (AP) propone un cambio di paradigma: l'unità base della computazione non è più il singolo dispositivo, ma un intero insieme (o aggregato) di nodi cooperanti, astraendo dettagli come il numero e l'esatta posizione dei singoli apparati [BPV15, BV16]. AP è un approccio di macro-programmazione funzionale basato sul concetto di campo computazionale (computational field), ovvero una struttura dati distribuita che mappa ogni dispositivo in un dato spazio e tempo a un valore [VAB+18, BPAFT25]. I dispositivi eseguono tutti un medesimo programma in modo asincrono (in round continui) seguendo un modello a tre fasi [BPAFT25]:

- *sense*: il nodo raccoglie informazioni dall'ambiente locale e dai messaggi ricevuti dai vicini durante la fase di riposo;
- *compute*: il programma aggregato viene valutato usando i dati raccolti, producendo un nuovo stato locale e i messaggi da condividere;
- *share*: il dispositivo invia i messaggi ai propri vicini.

2.2.2 Field Calculus (FC) e Building Blocks

Il fondamento matematico di Aggregate Programming è Field Calculus (FC), un calcolo universale minimale che modella le interazioni dei dispositivi nello spazio e nel tempo [BV16, VAB+18]. FC opera a un basso livello di astrazione, basandosi su un numero ridotto di costrutti [VAB+18]:

- `nbr` (*Neighborhood values*): permette l'interazione del dispositivo con il vicinato, raccogliendo e mappando i valori più recenti di una determinata espressione provenienti dai nodi vicini e dal nodo stesso [VAB+18, BV14];
- `rep` (*Time evolution*): modella la dinamica temporale, inizializzando una variabile di stato locale e aggiornandola ciclicamente a ogni round di computazione [VAB+18, BV14];
- `if` (*Domain restriction*): partiziona la rete in sottoregioni in cui vengono eseguiti rami di codice differenti [VAB+18].

Per aumentare il livello di astrazione ed evitare la gestione manuale dei costrutti come `rep` e `nbr`, come illustrato in fig. 2.1, sono stati identificati una serie di building blocks formalmente provati per essere auto-stabilizzanti. Questi blocchi incapsulano i costrutti di FC e ciò garantisce che il sistema che li utilizza sia in grado di recuperare da guasti e adattarsi ai cambiamenti topologici, convergendo sempre verso lo stato corretto [BV14, VAB+18]. I principali operatori di base sono [BV14, BPV15]:

- **G**: diffonde informazioni da una regione sorgente seguendo un gradiente (es. calcolo delle distanze minime o broadcast);
- **C**: accumula informazioni muovendosi lungo il gradiente di un potenziale (tipicamente verso un nodo pozzo o sink);
- **T**: regola l'evoluzione temporale, agendo come una memoria a tempo o un count-down;
- **S**: elegge dei leader locali in modo sparso, frammentando lo spazio in partizioni (es. diagrammi di Voronoi).

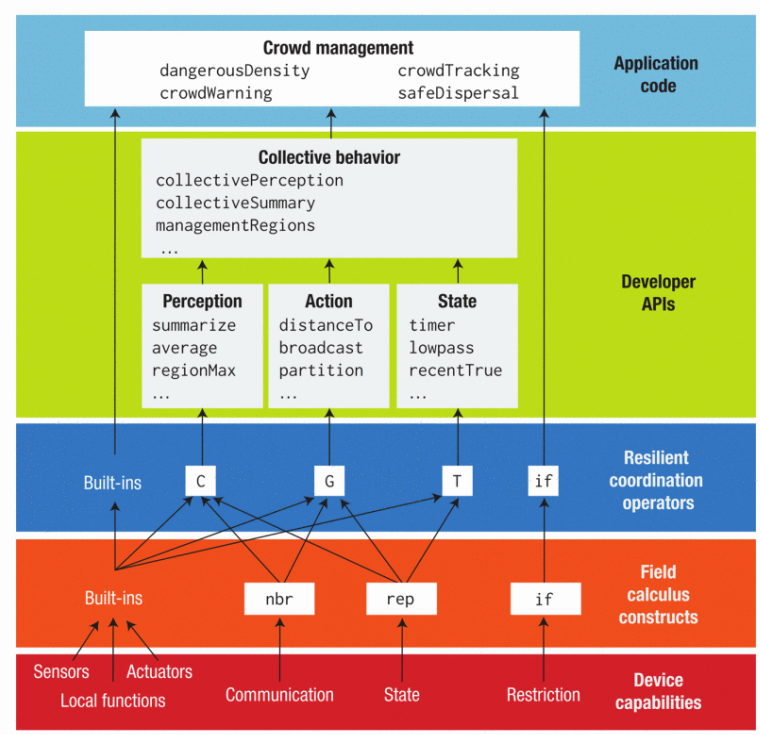

Figura 2.1: Livelli di astrazione della programmazione aggregata. Le capacità software e hardware di alcuni dispositivi sono utilizzate per implementare costrutti di FC a livello aggregato. Questi costrutti sono utilizzati per implementare operazioni di coordinamento a blocchi resilienti, che vengono poi racchiuse e combinate insieme per produrre API. © 2015 IEEE. Ristampata, con permesso, da [BPV15].

2.2.3 Self-organising Coordination Regions (SCR)

Sfruttando i building blocks, è stato ideato un design pattern chiamato SCR, pensato appositamente per gestire reti molto grandi e complesse, come quelle dell'Edge Computing [CPVN19]. Il suo obiettivo è trovare un compromesso efficace tra un controllo totalmente centralizzato e uno completamente decentralizzato [CPVN19]. Il sistema si auto-organizza suddividendosi spontaneamente in regioni. All'interno di ciascuna regione, le attività vengono coordinate in modo autonomo grazie a un continuo scambio di informazioni (un ciclo di feedback, come mostrato in fig. 2.2) tra un nodo responsabile (leader) e i dispositivi circostanti [CPVN19]. Il funzionamento di questo pattern si basa su quattro processi continui [CPVN19]:

1. *Elezione dei leader*: vengono eletti i leader da un insieme di candidati (tramite l'operatore S).

2. *Formazione delle regioni*: ogni nodo viene assegnato a un singolo leader, costruendo i percorsi di comunicazione ottimali.

3. *Raccolta delle informazioni*: flussi di dati si muovono dai nodi della regione verso il leader, utilizzando algoritmi come C.

4. *Propagazione delle informazioni*: i leader valutano i dati raccolti e diffondono i risultati o le direttive verso tutti i membri della propria regione (utilizzando G).

Il pattern SCR è resiliente: se un leader subisce un guasto o si verificano variazioni nel carico, il sistema riconfigura automaticamente i leader e ridimensiona le regioni in pochi secondi [CPVN19].

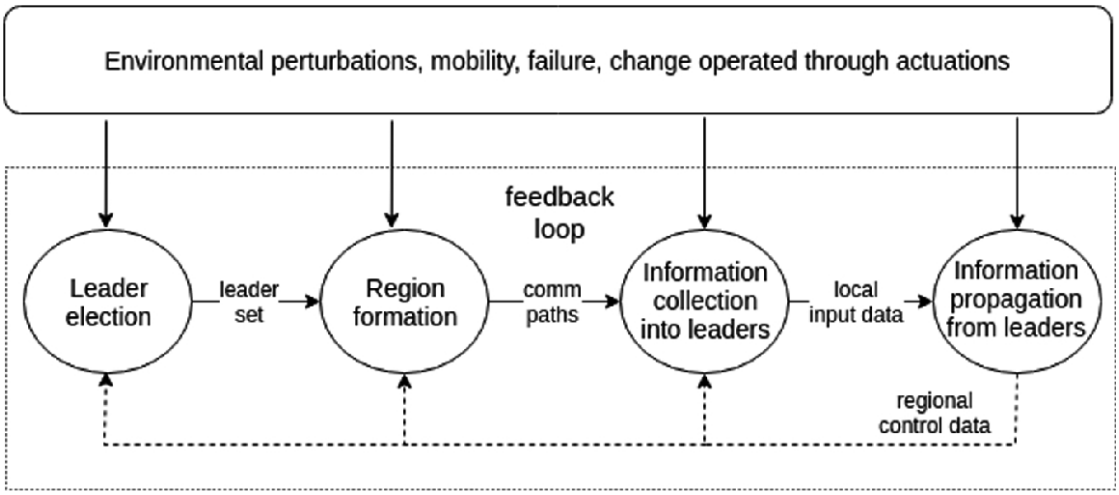

Figura 2.2: Prospettiva dinamica di SCR [CPVN19].

2.2.4 Collektive

Dal punto di vista dell'implementazione, la programmazione aggregata viene concretizzata tramite Domain Specific Languages (DSL), tra cui ScaFi[3] (basato su Scala) e Protelis[4] (basato su Java). In questa tesi viene utilizzato **Collektive**[5], un DSL nativo basato su Kotlin. È un framework open-source progettato per semplificare lo sviluppo di sistemi distribuiti attraverso l'utilizzo dei principi di AP [Col26]. Architetturalmente, Collektive è suddiviso in tre macro-componenti:

1. un plugin per il compilatore, che si occupa di gestire l'allineamento;
2. il DSL, che definisce i costrutti linguistici fondamentali;
3. e la libreria standard, che fornisce le funzioni essenziali per AP.

Il DSL Collektive espone funzioni per la gestione dello stato e delle interazioni di rete, ad esempio [BPAFT25]:

- `neighboring`: consente di osservare e mappare in un campo i valori di una specifica variabile nei nodi vicini. Corrisponde a nbr in FC;

[3] `https://scafi.github.io/`
[4] `https://protelis.github.io/wiki-beta/`
[5] `https://collektive.github.io`

- `exchange`: è uno degli operatori più importanti del DSL. Modella la comunicazione *anisotropica*, consentendo a un nodo di inviare messaggi differenti a vicini diversi.

- `share`: modella la comunicazione *isotropica* (invia lo stesso stato a tutti), combinando lo stato precedente con le informazioni scambiate nel vicinato per evolvere i dati nello spazio e nel tempo;

- `evolve`: equivale a `rep` nel FC e consente di modellare l'evoluzione dello stato del dispositivo nel tempo;

- `alignedOn`: introduce la segmentazione del dominio (domain segmentation), isolando le comunicazioni ed eseguendo i blocchi di codice esclusivamente tra i dispositivi che soddisfano e condividono un medesimo identificatore (pivot) [BPAFT25].

I building blocks vengono implementati componendo queste primitive. Operazioni come `gradientCast` o il calcolo delle distanze tramite Bellman-Ford si ottengono combinando `exchange` o `share` con una metrica, ad esempio calcolando la formula di Haversine sui dati GPS recuperati tramite `neighboring` [BPAFT25].

Per quanto concerne l'implementazione del pattern SCR, Collektive vi provvede utilizzando algoritmi di elezione e segmentazione. La formazione dinamica delle regioni è delegata alla funzione `boundedElection`, che partiziona la rete senza alcuna coordinazione centrale [BPAFT25]. L'algoritmo riceve in input il diametro massimo del cluster e una metrica di "forza" del nodo (la priorità del candidato). Questa "forza" coincide con l'eleggibilità del dispositivo a diventare leader (che può basarsi su risorse energetiche, di calcolo, o sul data rate come nel caso dell'esperimento iniziale di questa tesi [BPAFT25]). È possibile personalizzare l'elezione del leader utilizzando `Candidacy` (rappresenta la candidatura in un'elezione di un candidato) all'interno del parametro `selectBest` in `boundedElection`. Valutando le "forze", `boundedElection` risolve le contese ed elegge i leader, adattandosi tempestivamente ai guasti o ai cambiamenti topologici [BPAFT25]. Una volta eletti i leader, all'interno della regione viene utilizzato l'operatore `alignedOn(leader)`: in questo modo i dispositivi di una stessa regione restringono il proprio dominio alla sola regione di appartenenza, instaurando canali diretti (come dei `convergeCast`)

Listing 2.1: Esempio di codice Collektive per una semplice simulazione di programmazione aggregata.

```
fun Aggregate<Int>.temperatureEntrypoint(env: EnvironmentVariables,
    collektiveDevice: CollektiveDevice<*>): Boolean {
    val temp: Double = env["temperature"]
    val distances = with(collektiveDevice) { distances() }
    val isLeader = (boundedElection(bound = 20.0, metric = distances) == localId)
        .also { env["leader"] = it }
    val closestLeader = gradientCast(source = isLeader, local = localId, metric =
        distances)
    return alignedOn(closestLeader) {
        val regDistances = with(collektiveDevice) { distances() }
        val count = countDevices(sink = isLeader)
        val sum = convergeSum(sink = isLeader, local = temp)
        val avg = if (isLeader && count > 0) sum / count else 0.0
        val alarmDecision = isLeader && avg > 30.0
        gradientCast(source = isLeader, local = alarmDecision, metric =
            regDistances)
            .also { env["alarm"] = it }
    }
}
```

per instradare le informazioni senza interferire con le regioni adiacenti [BPAFT25]. `temperatureEntrypoint` (listing 2.1) rappresenta un esempio di un semplice algoritmo in cui vengono utilizzate le nozioni di SCR e AP: i nodi possiedono un attributo "temperatura". Essi si organizzano in regioni con il proprio leader e quest'ultimo si occupa di calcolare la media delle temperature dei dispositivi della propria area. Se la media supera i 30°C allora il leader diffonde un allarme.

2.3 Simulazione in ambito marittimo

2.3.1 Modeling & Simulation (M&S)

I porti marini e gli scenari di navigazione costiera sono sistemi altamente complessi, caratterizzati da un'elevata densità di traffico, condizioni meteo variabili e diversi attori (navi mercantili, piccoli natanti, rimorchiatori, e torri di controllo). In tale contesto, testare nuove configurazioni di rete, valutare l'efficienza delle operazioni o formare il personale sul campo risulta spesso proibitivo per questioni di sicurezza, limiti fisici ed elevati costi operativi [CPL15].

Per risolvere queste problematiche, il paradigma Modeling & Simulation (M&S) rappresenta lo strumento d'eccellenza. M&S consente di riprodurre il comportamento del sistema reale attraverso un modello computazionale, offrendo un ambiente virtuale in cui analizzare scenari "what-if", condurre analisi sugli incidenti e massimizzare l'efficacia del training [CPL15]. In particolare, la simulazione distribuita permette l'addestramento congiunto di piloti di navi e operatori del traffico (VTS) all'interno di un unico ambiente virtuale condiviso, riducendo i rischi e i costi associati agli errori in fase di apprendimento [CPL15]. Oltre alla formazione del personale, la simulazione si rivela oggi un mezzo fondamentale per lo sviluppo e la prototipazione di reti di telecomunicazione e schemi di coordinamento adattivo destinati alle future navi a guida autonoma (MASS) [BPAFT25, CPL15].

2.3.2 Alchemist

Per l'esecuzione e la valutazione dell'esperimento di questa tesi è stato adottato **Alchemist**[6], una piattaforma di simulazione di sistemi distribuiti, particolarmente adatta alla programmazione aggregata. Consente di modellare e simulare le interazioni tra agenti in ambienti complessi, supportando diverse astrazioni e paradigmi di programmazione [Col26].

[6] `https://alchemistsimulator.github.io/`

Meta-modello e Astrazioni

Per garantire la flessibilità necessaria a simulare scenari complessi, la logica di Alchemist, come mostrato in fig. 2.3, si basa su un meta-modello le cui entità fondamentali sono [Pia21]:

- *Environment*: lo spazio in cui vivono i nodi, con la responsabilità di mapparne la posizione. Può includere ostacoli fisici, mappe reali (es. file OpenStreetMap[7] o tracce GPS) e ambienti indoor [Pia21].
- *Node*: un contenitore situato nell'ambiente che ospita reazioni e "molecole".
- *Linking rule*: una funzione dell'ambiente che associa ogni nodo a un vicinato.
- *Molecule* e *Concentration*: la molecola rappresenta un dato, mentre la concentrazione ne rappresenta il valore associato all'interno di uno specifico nodo.
- *Reaction*: un evento che modifica lo stato dell'ambiente. È composto da condizioni (che ne determinano l'eseguibilità), una distribuzione temporale, un'equazione di tasso (*rate*) e azioni concrete (es. movimento del nodo o alterazione di una concentrazione) [PMV13, Pia21].

[7] `https://www.openstreetmap.org/`

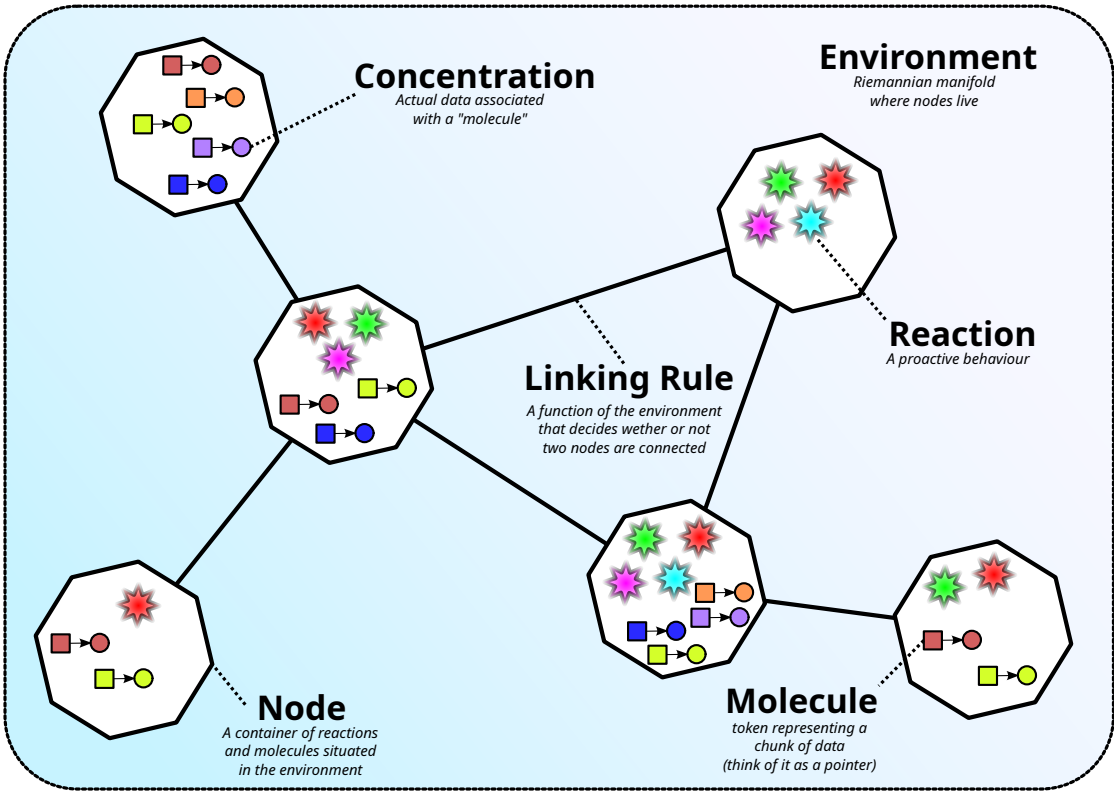

Figura 2.3: Rappresentazione grafica del meta-modello di Alchemist [Pia21].

Incarnazioni (*Incarnations*) per Aggregate Programming

Per supportare linguaggi e paradigmi differenti, Alchemist sfrutta il concetto di *Incarnation* (incarnazione), che permette di definire il tipo di dato esatto usato per le variabili (Concentrazione) e di fornire comportamenti specifici [Pia21]. Tra le incarnazioni supportate, assumono particolare rilevanza: `collektive`, `protelis` e `scafi`. Queste incarnazioni sono progettate specificamente per eseguire programmi basati su AP [Pia21]. Insieme, Alchemist e Collektive, consentono di implementare programmi aggregati in Kotlin che possono essere eseguiti e simulati in ambienti dinamici e configurabili [Col26].

Configurazione e Riproducibilità

Alchemist semplifica la configurazione degli ambienti di test affidandosi a specifiche scritte in *YAML*, un linguaggio di serializzazione dati in formato dichiarativo [Pia21]. Tramite un file YAML è possibile definire per esempio i modelli di rete (*network-model*) e la disposizione iniziale dei nodi (*deployments*). In listing 2.2 viene mostrata la struttura di una possibile configurazione di una simulazione, riprendendo l'esempio: listing 2.1. Un aspetto pratico fondamentale di Alchemist per le simulazioni in ambito marittimo è il supporto nativo a dati geospaziali. Il simulatore permette di importare ambienti fisici basati su OpenStreetMap e di impostare la dinamica dei nodi importando direttamente tracce GPS in formato GPS eXchange Format (GPX) (generate, ad esempio, dalla decodifica di messaggi AIS), consentendo di testare algoritmi sulle rotte effettivamente percorse dalle imbarcazioni [Pia21, BPAFT25]. Infine, Alchemist garantisce la ripetibilità degli esperimenti: il sistema separa e controlla in modo deterministico i seed (semi) dei generatori pseudo-casuali (come il Mersenne Twister) sia per la disposizione iniziale dei nodi, sia per l'esecuzione della simulazione [Pia21].

Listing 2.2: Parte della configurazione della simulazione per il programma: listing 2.1.

```
incarnation: collektive

network-model:
  type: ConnectWithinDistance
  parameters: [3.0]

_pool: &program
  - time-distribution: 1
    type: Event
    actions:
      - type: RunCollektiveProgram
        parameters: [it.unibo.collektive.TemperatureKt.temperatureEntrypoint]

deployments:
  - type: Grid
    parameters: [ 0, 0, 40, 80, 2, 2, 0.5, 0.5 ]
    programs:
      - *program
    contents:
      - molecule: leader
        concentration: false
      - molecule: temperature
        concentration: 17.0
```

2.4 Algoritmo precedente

L'esperimento su cui si basa questa tesi ha come obiettivo quello di migliorare la raccolta e la trasmissione di dati in uno scenario marittimo realistico. Il caso di studio considera il fiordo di Kiel, in Germania, dove le navi sono modellate come nodi in grado di comunicare tra loro e con stazioni a terra [BPAFT25]. Il problema affrontato nasce dal divario tra la quantità di dati generata dai moderni sistemi di navigazione autonoma (MASS) e la capacità delle infrastrutture di comunicazione marittima tradizionali. Navi autonome o semi-autonome possono infatti produrre flussi di dati ad alta intensità, derivanti ad esempio da telecamere e LiDAR; al contrario, tecnologie tradizionali come AIS e sistemi basati su VHF offrono una capacità trasmissiva limitata [BPAFT25]. Nella simulazione, ogni imbarcazione è dotata di diversi dispositivi di comunicazione: un sistema Automatic Packet Reporting System (APRS) su VHF, un dispositivo Long Range Wide Area Network (LoRaWAN) di classe C, un modulo 5G di tipo consumer e un dispositivo compatibile con Wi-Fi Direct [BPAFT25]. Inoltre, alcune imbarcazioni possono essere equipaggiate con una torre 5G montata a bordo, con probabilità indicata

dal parametro p_{5G}. Ogni nave tenta di trasmettere verso terra dei dati di dimensione fissata pari a $d = 3$ Mbit/s, valore scelto come approssimazione del bitrate necessario per un flusso video compresso a 720p [BPAFT25].

Per stimare la qualità dei collegamenti, l'esperimento associa a ogni coppia di nodi un data rate stimato in funzione della tecnologia disponibile e della distanza tra i dispositivi. Le tecnologie considerate sono quindi 5G, Wi-Fi, LoRaWAN e APRS. A partire da valori di riferimento ricavati dalla letteratura, il data rate in linea di vista viene stimato mediante interpolazione log-lineare rispetto alla distanza [BPAFT25].

Gli approcci confrontati nell'esperimento precedente sono quattro:

- *Baseline non cooperativa*: rappresenta lo stato dell'arte di riferimento. Ogni nave tenta di comunicare direttamente con una stazione a terra utilizzando la tecnologia più favorevole tra quelle disponibili. Se il collegamento diretto offre un data rate sufficiente a trasmettere il flusso d, la comunicazione viene considerata riuscita; in caso contrario, il flusso viene scartato [BPAFT25]. Questo approccio non prevede alcuna collaborazione tra imbarcazioni.

- *Distance-based Multi-Relay Communication (Dist-MR)*: ogni nave che non riesce a raggiungere direttamente la stazione a terra seleziona come relay un'imbarcazione vicina lungo il cammino geograficamente più breve verso la destinazione. Si ha quindi un insieme di alberi che originano nelle stazioni a terra. La metrica usata per la scelta del relay è la distanza geografica [BPAFT25].

- *Data Rate-based Multi-Relay Communication (DR-MR)*: modifica il criterio di scelta del relay utilizzato in Dist-MR. Invece di minimizzare la distanza geografica, l'obiettivo è selezionare il percorso che massimizza il data rate disponibile verso la stazione a terra [BPAFT25].

- *Collective Summarisation Clusters (CSC)*: organizza le imbarcazioni in cluster auto-organizzanti, la cui logica è implementata sfruttando la programmazione aggregata tramite il DSL Collektive. Ogni cluster possiede un leader, responsabile della raccolta dei flussi dati prodotti dai membri della regione e della loro successiva sintesi prima dell'invio verso terra o verso un

altro cluster. Questo approccio si basa sul pattern SCR: i leader vengono eletti dinamicamente e la formazione dei cluster tiene conto della capacità di trasmettere dati verso il leader [BPAFT25].

Capitolo 3

Analisi

3.1 Limiti dell'algoritmo esistente

L'algoritmo analizzato nel capitolo precedente (sezione 2.4) organizza le navi in cluster valutandone principalmente la posizione e il canale di comunicazione con il leader. In particolare, vengono utilizzati la longitudine e la latitudine delle imbarcazioni e il data rate [BPAFT25]. A sua volta, il leader comprime i flussi raccolti e inoltra l'output sintetizzato a un relay, ovvero un nodo intermedio utilizzato per stabilire percorsi di comunicazione multi-hop verso la stazione a terra o il cluster successivo [BPAFT25]. Questo approccio non tiene conto delle dinamiche complesse che caratterizzano l'ambiente marittimo. Se in un dato istante due nodi sono fisicamente vicini, potrebbero non esserlo più poco tempo dopo a causa della loro elevata differenza di rotta o velocità. Raggrupparli in uno stesso cluster comporterebbe una continua variazione della topologia di rete, il che innescherebbe continue rielezioni del leader o dei relay e frammentazioni della rete.

Dunque, il principale limite dell'algoritmo attuale risiede nel mancato sfruttamento dei dati a disposizione nel payload AIS. Come illustrato nella sezione 2.1.2, i messaggi del Sistema di Identificazione Automatica (in particolare quelli di tipo 1, 2 e 3) contengono diverse informazioni di navigazione, tra cui: SOG (velocità), COG (direzione) e ROT (velocità di virata) [Int26]. Al momento, l'algoritmo ignora completamente questi dati, limitandosi a estrarre solo la posizione.

Oltre a ciò, l'esperimento presenta ulteriori limitazioni. In particolare, le valu-

tazioni si basano su una quantità modesta di dati. I dati utilizzati per l'esperimento provengono da uno scenario reale nel fiordo di Kiel. Tuttavia, riguardano una finestra circoscritta a sei ore nell'arco di una giornata, che potrebbe non essere rappresentativa del traffico nel fiordo [BPAFT25]. Inoltre, lo studio non ha esaminato casi limite specifici (come i guasti al nodo leader) per esaminare il comportamento degli approcci valutati in particolari situazioni, né ha preso in considerazione le avversità e il mutare delle condizioni meteo marine [BPAFT25]. Infine, non si è tenuto conto degli aspetti sulla sicurezza trattati nel primo punto della sezione 2.1.4 e della ricerca di algoritmi di sintesi dati [BPAFT25].

3.2 Obiettivi

L'obiettivo di questa tesi (in linea con le direttive della Fase 2 della roadmap *CoMPass* [AFBPT25]) è quello di superare i limiti dell'approccio che tiene conto solo della posizione, analizzando come la creazione dei cluster tenendo conto della direzione e della velocità delle imbarcazioni possa influire sulla stabilità della rete. Nello specifico, si intende estendere l'algoritmo CSC estraendo dal payload AIS i dati di SOG (velocità rispetto al fondo) e COG (rotta rispetto al fondo) per integrarli direttamente all'interno dell'algoritmo. Due navi che, in un determinato istante, risultano geograficamente vicine e presentano un buon data rate possono essere assegnate alla stessa regione. Tuttavia, se tale scelta non considera le differenze di direzione e velocità, le imbarcazioni potrebbero allontanarsi rapidamente rendendo necessarie frequenti riorganizzazioni della rete. In base a quanto le imbarcazioni viaggiano in direzioni simili e velocità affini, si punta a creare cluster più stabili nel tempo, riducendo la volatilità della topologia della rete per valutare se tale stabilità possa tradursi in una maggiore efficienza della comunicazione verso la costa.

3.3 Requisiti

Per garantire il raggiungimento degli obiettivi prefissati, il nuovo algoritmo deve soddisfare una serie di requisiti. Tali requisiti sono elencati di seguito suddivisi in

funzionali (le operazioni che il sistema deve compiere) e non funzionali (come le deve compiere).

3.3.1 Requisiti funzionali

- Il sistema deve poter estrarre dai messaggi AIS la velocità e la rotta.
- In assenza delle informazioni cinematiche, deve essere possibile continuare a operare senza tali dati.
- L'algoritmo deve valutare le differenze di SOG e COG tra imbarcazioni e favorire, nella scelta dei relay o nella formazione dei cluster, nodi con una mobilità compatibile con la nave presa in esame.
- La simulazione deve essere in grado di esporre i dati ottenuti per poter effettuare valutazioni e confronti con la versione precedente dell'algoritmo.

3.3.2 Requisiti non funzionali

- Il calcolo delle nuove metriche con l'aggiunta di SOG e COG non deve introdurre un carico computazionale tale da far perdere i vantaggi del CSC.

Capitolo 4

Design e implementazione

4.1 Strategia di integrazione nel criterio di clustering

La soluzione proposta (visionabile nel repository dell'esperimento[1]) introduce una penalità di mobilità basata su due componenti:

- Componente angolare (COG): misura la differenza di direzione tra due nodi. Calcola l'angolo più breve tra le due rotte, normalizzato nell'intervallo $[0, 1]$, dove 0 indica nodi che si muovono nella stessa direzione e 1 indica nodi che si muovono in direzioni opposte.
- Componente di velocità (SOG): misura il divario della velocità di movimento tra due navi rapportato a una soglia massima tollerata di 25 nodi. Anche questo valore è normalizzato in $[0, 1]$.

Penalità unificata: Le due componenti vengono combinate in una penalità di mobilità unica tramite una media ponderata, dove la componente angolare riceve un peso maggiore (60%) rispetto a quella di velocità (40%).

Metrica finale: La penalità di mobilità viene applicata alla metrica originale di trasmissione attraverso un fattore moltiplicativo. In questo modo, i percorsi

[1] `https://doi.org/10.5281/zenodo.20810186`

tra nodi con mobilità molto differente vengono penalizzati, aumentando il costo computazionale degli algoritmi di elezione del leader e selezione dei relay.

L'introduzione della metrica consapevole della mobilità influenza diversi contesti dell'algoritmo di clustering. In primis, questo valore incide sul calcolo della distanza topologica verso la stazione base, fondamentale per l'elezione del leader. In secondo luogo, la nuova metrica interviene nella misurazione delle distanze intra-cluster tra leader e nodi membri e nella selezione dei relay inter-cluster per il routing dei dati. Questo garantisce che siano preferiti nodi con mobilità simile per mantenere le comunicazioni.

4.2 Estrazione e integrazione delle feature AIS aggiuntive

I parametri principali estratti dai messaggi AIS includono: l'ID della nave, la posizione geografica (longitudine e latitudine), il timestamp del messaggio, SOG e COG. Nel formato AIS raw NMEA 0183 utilizzato in questo lavoro, tali valori sono codificati secondo le specifiche standard: SOG è espresso come valore intero in decimi di nodo (utilizzando il valore 1023 per indicare un dato non disponibile), mentre COG è codificato in decimi di grado, usando 3600 (equivalente a 360.0 gradi) per indicare "dato non disponibile" [Int26]. Durante la fase di parsing, questi valori vengono decodificati dividendo per i rispettivi fattori di scala per ottenere i valori in unità standard (nodi e gradi).

4.2.1 Decodifica messaggi AIS

Per la decodifica dei messaggi AIS, è stata utilizzata la libreria `dk.dma.ais`, sviluppata dal Danish Maritime Authority (DMA) [Dan]. Questa libreria fornisce un parser in grado di gestire il protocollo NMEA 0183 standard e di decodificare messaggi AIS multi-riga.

La libreria è integrata nel progetto e offre le seguenti funzionalità [Dan]:

- parsing dei singoli frame AIS secondo le specifiche NMEA 0183;
- ricomposizione automatica dei messaggi multi-sentenza;

Listing 4.1: Parte del codice della classe `AisPayload`.

```
data class AisPayload(
    val boatId: Int,
    val timestamp: Instant,
    val longitude: Double,
    val latitude: Double,
    val speedOverGround: Double?,
    val courseOverGround: Double?,
) {
    companion object {
        fun from(
            boatId: Int,
            timestamp: Instant,
            aisMessage: AisMessage,
        ): AisPayload? =
            if (
                aisMessage is IPositionMessage &&
                ParsingUtils.validateLongitude(aisMessage.pos.longitudeDouble) &&
                ParsingUtils.validateLatitude(aisMessage.pos.latitudeDouble)
            ) {
                // Se messaggio non contiene sog e cog, questi diventano null
                val vesselPositionMessage = aisMessage as? IVesselPositionMessage
                AisPayload(
                    boatId,
                    timestamp,
                    aisMessage.pos.longitudeDouble,
                    aisMessage.pos.latitudeDouble,
                    if (vesselPositionMessage?.isSogValid == true)
                        vesselPositionMessage.sog / 10.0 else null,
                    if (vesselPositionMessage?.isCogValid == true)
                        vesselPositionMessage.cog / 10.0 else null,
                )
            } else {
                null
            }
    }
}
```

- validazione del checksum;
- decodifica dei campi contenuti nei messaggi.

Nel caso qui considerato (listing 4.1), la libreria fornisce accesso ai campi di velocità e rotta attraverso l'interfaccia `IVesselPositionMessage`, che rappresenta i messaggi AIS contenenti informazioni di posizionamento (tipicamente messaggi di tipo 1, 2, 3) [Dan]. Si controlla se il messaggio attuale contenga SOG e COG e, se non li contiene, viene assegnato loro il valore `null`.

Listing 4.2: Estensione aggiunta alle tracce GPX delle navi.

```
<extensions>
    <sog>XXX.X</sog>
    <cog>XXX.X</cog>
</extensions>
```

4.2.2 Generazione tracce GPX

Successivamente alla conversione dei dati AIS, SOG e COG vengono integrati all'interno dei file GPX generati dai messaggi del Sistema di Identificazione Automatica. Le tracce GPX fungono da ponte tra i dati AIS e la simulazione Alchemist, consentendo di riprodurre i movimenti delle imbarcazioni e allegare metadati aggiuntivi ai nodi.

Nella fase di generazione delle tracce, sono state apportate le seguenti modifiche:

1. *Anonimizzazione del nome*: i nomi dei vascelli (MMSI) vengono sostituiti con identificativi anonimizzati nel formato `anonXXXX`, dove `XXXX` è un numero progressivo.

2. *Estensioni GPX*: Per ogni punto di traccia (`trkpt`), sono stati aggiunti elementi di estensione contenenti SOG e COG nel formato indicato nel listing 4.2.

4.2.3 Caricamento di SOG e COG nella simulazione

Durante la simulazione Alchemist, l'azione `LoadSogCogFromTrace` (listing 4.3) è responsabile della lettura di SOG e COG dai file GPX e della loro disponibilità come molecole nel simulatore.

L'azione funziona nel seguente modo:

1. associa ogni nodo della simulazione al corrispondente file GPX in base all'indice;

2. per ogni passo di simulazione, individua il punto di traccia corrispondente al tempo di simulazione corrente;

Listing 4.3: Codice semplificato dell'azione Alchemist per leggere SOG e COG dalle tracce GPX.

```
class LoadSogCogFromTrace<T>(
    private val environment: NavigationEnvironment<T>,
    node: Node<T>,
    private val path: String,
    referenceTimeEpochSeconds: Number,
) : AbstractAction<T>(node) {

    // Inizializzazione dei percorsi e filtro dei file .gpx omessi

    private val points by lazy {
        val nodeIndex = environment.nodes.indexOf(node)
        // Associazione del nodo al file traccia corrispondente
        readPoints(files[nodeIndex])
    }

    override fun execute() {
        val time = environment.simulation.time.toDouble()
        // Ricerca dell'ultimo punto rispetto al tempo di simulazione
        val point = points.lastOrNull { it.time <= time } ?: points.firstOrNull()

        // Aggiornamento delle molecole nel nodo (Speed e Course Over Ground)
        point?.sog?.takeUnless(Double::isNaN)?.let { set("sog", it) }
        point?.cog?.takeUnless(Double::isNaN)?.let { set("cog", it) }
    }

    // Metodi di supporto e data class Point omessi
}
```

3. legge i valori di SOG e COG dalla traccia;
4. filtra i valori non disponibili (`NaN`);
5. assegna i valori come concentrazioni delle molecole `sog` e `cog` nel nodo (listing 4.4).

Listing 4.4: Utilizzo di LoadSogCogFromTrace nel file di configurazione della simulazione.

```
_pools:
  - pool: &move
      - time-distribution: 0.1
        type: Event
        actions:
          - type: LoadSogCogFromTrace
            parameters:
              path: "navigation-routes"
              referenceTimeEpochSeconds: *start-time
```

4.2.4 Integrazione dei dati AIS in Alchemist

Sebbene per l'esecuzione di questo esperimento ci si sia avvalsi della struttura basata sul pre-processamento dei file GPX, il lavoro di tesi si è spinto oltre per superare questo limite strutturale. Ho introdotto il supporto nativo per i dati AIS nel simulatore Alchemist, visionabile a questo link[2]. I dati AIS vengono ora integrati attraverso un loader dedicato (`AISTraceLoader`) che consente il caricamento di file NMEA e AIS. Il codice estrae diverse informazioni dai messaggi decodificati, includendo i dati presenti nella tabella 2.1. Questa integrazione permette di rappresentare ciascuna nave come un'entità indipendente tramite la proprietà di nodo `AISVessel`.

4.3 Modifiche all'algoritmo

4.3.1 Costanti definite e lettura dei valori

I valori di SOG e COG vengono letti dall'ambiente di simulazione tramite la funzione `getIfDefined`, che restituisce `NaN` se il parametro non è disponibile.

Il listing 4.5 mostra le costanti utilizzate per il calcolo della penalità di mobilità e la funzione `getIfDefined`. Questi valori sono stati scelti in modo euristico per bilanciare l'influenza della direzione e della velocità sulla metrica di selezione.

[2] `https://github.com/AlchemistSimulator/Alchemist/tree/master/alchemist-maps`

Listing 4.5: Costanti utilizzate per il calcolo della penalità di mobilità.

```
private const val COG_WEIGHT = 0.6
private const val SOG_WEIGHT = 0.4
private const val MAX_TOLERATED_SPEED_DIFF_KNOTS = 25.0
private const val MOBILITY_PENALTY_MULTIPLIER = 2.0
private const val HALF_TURN_DEGREES = 180.0
private const val FULL_TURN_DEGREES = 360.0

/** Reads an optional Double property from the environment. */
private fun CollektiveDevice<*>.getIfDefined(name: String): Double =
    if (isDefined(name)) get<Double>(name) else Double.NaN
```

Significato dei parametri

- `COG_WEIGHT` e `SOG_WEIGHT`: Il peso della direzione e della velocità nel calcolo della penalità combinata. Si è deciso di attribuire più importanza alla direzione (0.6) che alla velocità (0.4).
- `MAX_TOLERATED_SPEED_DIFF_KNOTS`: La differenza massima di velocità tollerata (25 nodi). Differenze superiori a questo valore sono considerate completamente penalizzanti.
- `MOBILITY_PENALTY_MULTIPLIER`: Un fattore di amplificazione (2.0) che rende la penalità di mobilità significativa nel calcolo della metrica finale.

4.3.2 Funzioni di calcolo della penalità

Penalità angolare (COG)

Il listing 4.6 implementa il calcolo della penalità angolare, che quantifica quanto la direzione di un nodo differisce da quella di un suo vicino:

L'algoritmo calcola la minima distanza angolare tra due direzioni, considerando che gli angoli sono circolari ([0°, 360°)). Ad esempio, una nave a 350° e una a 10° hanno una differenza di soli 20° e non di 340°. Il risultato è normalizzato in [0, 1], dove 0 indica navi perfettamente allineate e 1 indica navi con direzioni opposte.

Penalità di velocità (SOG)

Il listing 4.7 implementa il calcolo della penalità di velocità:

Listing 4.6: Calcolo della penalità angolare basata su COG.

```
private fun angularMobilityPenalty(
    sourceCog: Double,
    targetCog: Double,
): Double = when {
    sourceCog.isNaN() || targetCog.isNaN() -> 0.0
    else -> {
        val diff = abs(sourceCog - targetCog) % FULL_TURN_DEGREES
        val angularDistance = if (diff > HALF_TURN_DEGREES) FULL_TURN_DEGREES -
            diff else diff
        angularDistance / HALF_TURN_DEGREES
    }
}
```

Listing 4.7: Calcolo della penalità di velocità basata su SOG.

```
private fun speedMobilityPenalty(
    sourceSog: Double,
    targetSog: Double,
): Double = when {
    sourceSog.isNaN() || targetSog.isNaN() -> 0.0
    else -> (abs(sourceSog - targetSog) / MAX_TOLERATED_SPEED_DIFF_KNOTS).coerceIn
        (0.0, 1.0)
}
```

Questa funzione calcola la penalità come il rapporto tra la differenza assoluta di velocità tra i due nodi e una soglia di tolleranza. La penalità è limitata a 1,0 se la differenza eccede la soglia, poiché differenze maggiori non fornirebbero informazioni aggiuntive utili. Il risultato è quindi sempre in $[0, 1]$, coerentemente con la penalità angolare descritta nella sezione precedente (sezione 4.3.2).

4.3.3 Integrazione nella metrica di selezione

Calcolo della penalità combinata

Il listing 4.8 mostra le modifiche effettuate nell'entry point della simulazione. Vengono lette la velocità e la direzione della nave attuale dalla simulazione e si ottiene tramite `neighboring` un campo contenente SOG e COG del proprio vicinato. In seguito, tramite le funzioni mostrate nella sezione 4.3.2 e nella sezione 4.3.2, vengono calcolate le penalità. Queste ultime vengono unite e ponderate (dando più importanza alla direzione) e viene calcolata la metrica finale `mobilityAwareMetric`. La metrica finale è ottenuta moltiplicando il tempo di trasmissione originale per

Listing 4.8: Modifiche inserite nell'algoritmo della simulazione.

```
val mySog = environment.getIfDefined("sog")
val myCog = environment.getIfDefined("cog")
val neighborSogs = neighboring(mySog)
val neighborCogs = neighboring(myCog)

// Calcolo delle penalizzazioni per ogni vicino
val angularPenalty = neighborCogs.map { neighborCog ->
    angularMobilityPenalty(myCog, neighborCog.value)
}
val speedPenalty = neighborSogs.map { neighborSog ->
    speedMobilityPenalty(mySog, neighborSog.value)
}

// Combinazione pesata delle penalizzazioni
val combinedMobilityPenalty = angularPenalty.alignedMapValues(speedPenalty) {
    cogPen, sogPen ->
    (cogPen * COG_WEIGHT) + (sogPen * SOG_WEIGHT)
}.inject(environment, "mobility-penalty")

// Applicazione della penalizzazione alla metrica originale
val mobilityAwareMetric = timeToTransmit.alignedMapValues(combinedMobilityPenalty)
     { time, penalty ->
    time * (1.0 + (penalty * MOBILITY_PENALTY_MULTIPLIER))
}.inject(environment, "mobility-aware-metric")
```

un fattore che incorpora la penalità di mobilità. Quindi, nel caso peggiore, per due navi con direzioni opposte e velocità molto diverse, la metrica arriva a tre volte `timeToTransmit`. Se invece non ci sono differenze significative, oppure se SOG e COG non sono disponibili, la metrica resta equivalente a `timeToTransmit`.

Listing 4.9: Utilizzo della nuova metrica nella selezione del leader e del relay.

```
// Calcolo della distanza al ground station
val clusteredTimeToStation: Double =
    distanceTo(
        groundStation,
        metric = mobilityAwareMetric,
    ).inject(environment, "clustered-timeToStation")

// Elezione del leader di cluster
val myLeader: Int =
    boundedElection(
        strength = -clusteredTimeToStation,
        bound = streamingBitRate.timeToTransmitOneMb,
        metric = mobilityAwareMetric,
        selectBest = { c1, c2 ->
            maxOf(
                c1,
                c2,
                compareBy<Candidacy<Int, Double, Double>> { it.strength }.thenBy { it.candidate },
            )
        },
    ).inject(environment, "myLeader")

// Selezione del relay per la trasmissione
val myRelay =
    potentialRelays
        .alignedMapValues(timesToStationAround + mobilityAwareMetric) { canRelay, distance ->
            when {
                canRelay -> distance
                else -> POSITIVE_INFINITY
            }
        }.all
        .fold(localId to POSITIVE_INFINITY) { current, entry ->
            if (entry.value < current.distanceToLeader) entry.id to entry.value else current
        }.relayId()
        .inject(environment, "myRelay")
```

Applicazione nei calcoli

La metrica consapevole della mobilità è applicata principalmente in tre punti dell'algoritmo (listing 4.9):

La metrica è usata come funzione di costo nell'operatore `distanceTo`, che calcola il gradiente di distanza da ogni nodo verso il ground station. Inoltre è utilizzata per l'elezione dei leader dei cluster e per la scelta dei relay.

Capitolo 5

Valutazioni

In questo capitolo vengono presentati e confrontati i risultati ottenuti dalla sperimentazione precedente e da quella attuale.

La metrica principale utilizzata per confrontare gli approcci è il data rate effettivamente ottenuto da ciascuna imbarcazione nella comunicazione verso la stazione a terra. Nel caso della baseline, tale valore coincide con il data rate del collegamento diretto. Invece per Dist-MR e DR-MR si segue il Minimum Spanning Tree (MST) verso la stazione a terra e si verifica, a ogni hop, se la capacità residua del collegamento consente di inoltrare sia il flusso locale sia quelli provenienti dai figli [BPAFT25]. Indicando con DR_n^u il data rate disponibile in uscita dal nodo n, con C il numero di figli e con d il data stream locale, il data rate disponibile per ciascun figlio viene calcolato come:

$$DR_n^f = \frac{DR_n^u - d}{C} \tag{5.1}$$

Il flusso viene trasmesso se $DR_n^f > d$ [BPAFT25].

Per l'approccio CSC vengono introdotte ulteriori metriche. La principale è il *reduction factor* RF, che misura il grado di compressione, riduzione o sintesi richiesto al leader per inoltrare i dati aggregati verso il relay successivo o verso la stazione a terra. Per un cluster composto da C nodi, con data rate in uscita del leader pari a DR^u e data stream individuale pari a d, il fattore di riduzione è definito come [BPAFT25]:

$$RF = \min\left(1, \frac{DR^u}{dC}\right) \tag{5.2}$$

Oltre al fattore di riduzione, vengono considerati la dimensione media dei cluster, il numero di cluster costituiti da una sola nave (*singleton clusters*) e il numero totale di cluster generati [BPAFT25].

5.1 Risultati dell'esperimento precedente

L'esperimento precedente è stato ripetuto cento volte utilizzando seed casuali differenti, i quali controllano sia quali navi vengono equipaggiate con torri 5G, sia la schedulazione dei round di calcolo di AP [BPAFT25]. Una delle misurazioni ha confrontato il data rate medio (espresso in Kbps) ottenuto dalla baseline non cooperativa con quello dei vari approcci multi-relay (Dist-MR, DR-MR e CSC), al variare della frequenza di aggiornamento e della probabilità di presenza della connessione 5G [BPAFT25]. Come illustrato in fig. 5.1, dotare le navi di funzionalità di backhaul tramite torri 5G (p_{5G}) ha un grande potenziale per migliorare i tassi di trasmissione dei dati [BPAFT25]. Tuttavia, la baseline e gli approcci Dist-MR e DR-MR mostrano miglioramenti limitati. Al contrario, un'implementazione anche modesta ($p_{5G} = 0.01$) produce miglioramenti considerevoli per CSC, che supera costantemente tutte le altre strategie, dimostrando picchi più elevati e data rate sostenuti durante l'intera simulazione [BPAFT25]. Sorprendentemente, l'approccio DR-MR offre prestazioni generalmente inferiori. Questo risultato controintuitivo è presumibilmente dovuto al fatto che le navi con un buon data rate vengono selezionate in modo eccessivo, finendo per dover gestire troppi flussi e causando la congestione di pochi canali [BPAFT25].

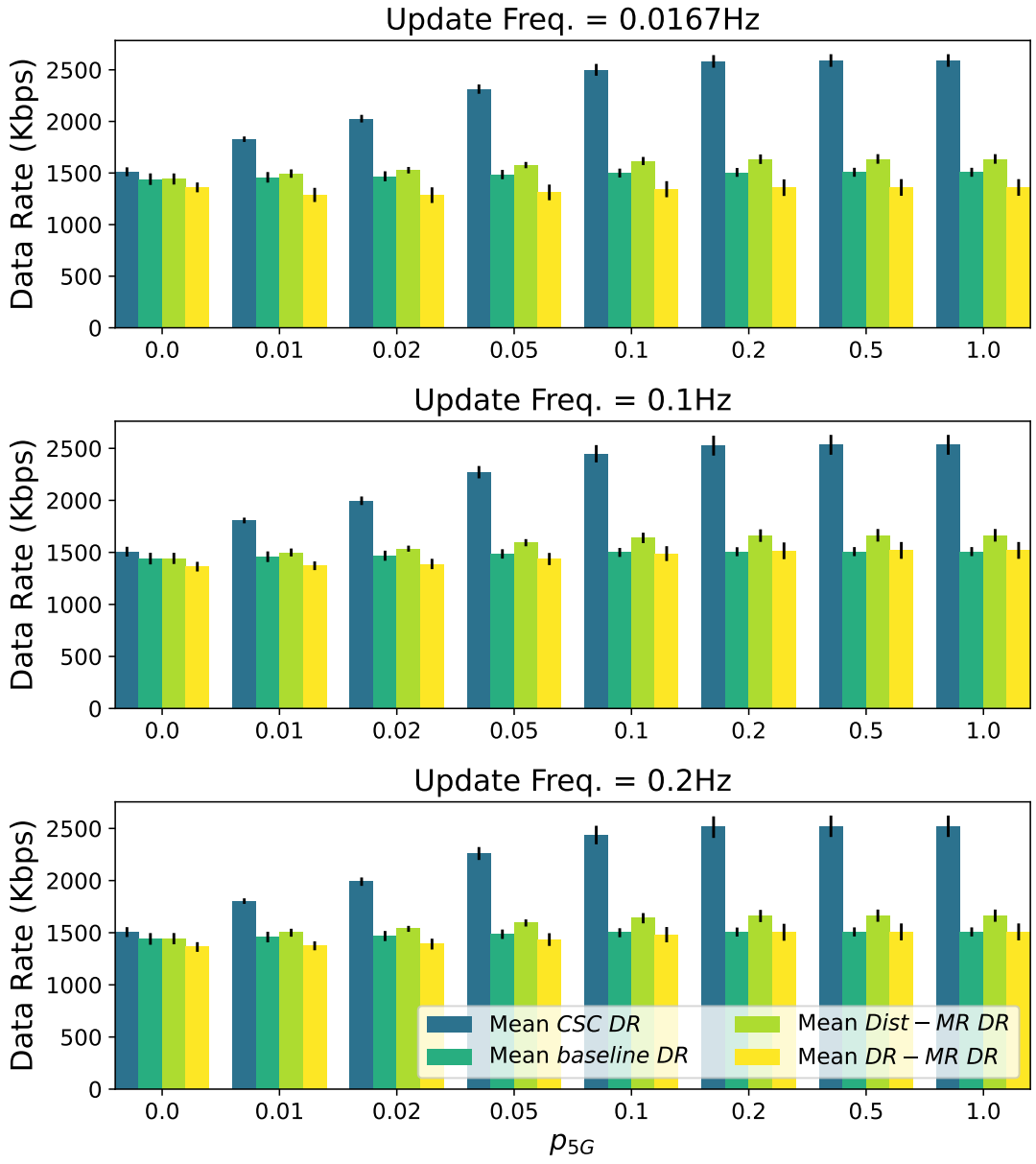

Figura 5.1: Confronto del data rate medio al variare della probabilità di equipaggiamento 5G delle navi (p_{5G}), della frequenza di esecuzione del round (f) e dell'algoritmo utilizzato. La presenza di navi dotate di 5G migliora notevolmente la velocità di trasmissione complessiva. L'approccio CSC risulta il più performante in ogni condizione valutata, sfruttando al meglio i benefici derivanti dal 5G [BPAFT25].

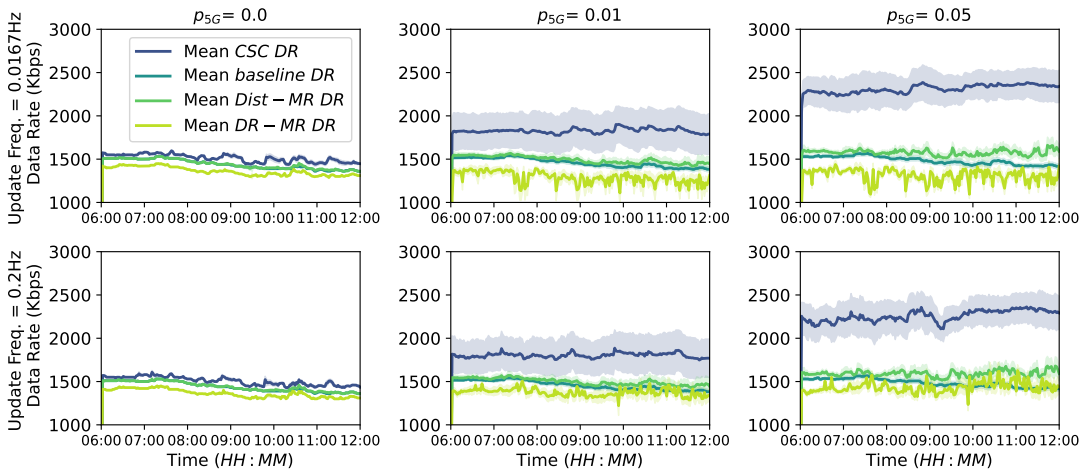

Figura 5.2: Andamento temporale del data rate medio per diverse probabilità di imbarcazioni equipaggiate con torri 5G (p_{5G}), confrontando i diversi algoritmi e le diverse frequenze di aggiornamento. La linea continua indica il valore medio, mentre le aree ombreggiate rappresentano l'intervallo di 1σ attorno alla media. Anche una presenza minima di imbarcazioni con tecnologia 5G porta a un miglioramento significativo delle prestazioni. L'approccio CSC risulta costantemente superiore agli altri in termini di velocità di trasmissione, sfruttando in modo più efficace la disponibilità del 5G [BPAFT25].

Per quanto riguarda l'evoluzione temporale, la fig. 5.2 dimostra che la frequenza di esecuzione di AP (f) ha un impatto molto modesto sulle prestazioni [BPAFT25]. Frequenze più elevate aiutano a ridurre la varianza (evidenziata dalle ombreggiature attorno alle medie nei plot temporali), ma anche a basse frequenze si osservano risultati stabili, indicando che l'overhead sul canale generato dagli algoritmi auto-organizzanti è trascurabile nella maggior parte dei casi [BPAFT25].

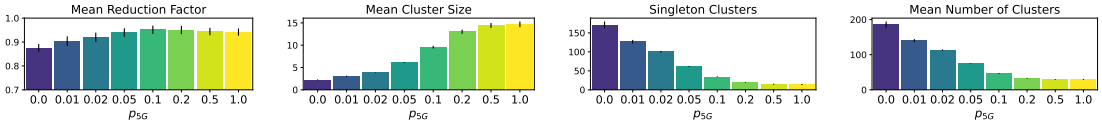

Figura 5.3: Metriche specifiche di CSC da sinistra verso destra: fattore di riduzione medio, dimensione media del cluster, numero di cluster singoli e numero medio di cluster. Tutti i dati sono stati misurati per $f = 0,2$ Hz e per diverse probabilità di torri 5G installate sulle imbarcazioni (p_{5G}) [BPAFT25].

Infine, l'analisi delle metriche specifiche per CSC (si veda fig. 5.3) rivela come l'algoritmo reagisce all'infrastruttura. Il fattore di riduzione medio aumenta con p_{5G}, avvicinandosi al valore 1. Questo indica che i leader dei cluster possono inoltrare una porzione maggiore dei flussi originali con perdite o sintesi minime [BPAFT25]. Le prestazioni del fattore di riduzione raggiungono un picco a $p_{5G} = 0.2$ per poi diminuire; l'ipotesi formulata dagli autori è che, quando i cluster diventano troppo grandi, si verifichino fenomeni di congestione per i relay intra-cluster simili a quelli osservati in DR-MR [BPAFT25]. Di conseguenza, all'aumentare di p_{5G}, la dimensione media dei cluster cresce (da 1-2 navi fino a più di 10), il numero di *singleton clusters* diminuisce drasticamente e il numero totale di cluster si riduce, poiché la formazione di cluster più grandi diventa sempre più conveniente [BPAFT25].

5.2 Risultati del nuovo algoritmo

La simulazione con le modifiche apportate nel capitolo 4 introduce una nuova variante dell'algoritmo CSC, denominata *Mobility-aware Collective Summarisation Clusters (M-CSC)*. L'esperimento è stato ripetuto dieci volte con seed casuali diversi. Il primo risultato riguarda il confronto del data rate medio ottenuto dai vari algoritmi dell'esperimento. Come si osserva dalla fig. 5.4, il 5G migliora considerevolmente la trasmissione dati dell'approccio M-CSC, evidenziando un data rate nettamente più alto rispetto agli altri metodi. Invece nella baseline, nel Dist-MR e nel DR-MR l'apporto del 5G non ha portato a miglioramenti.

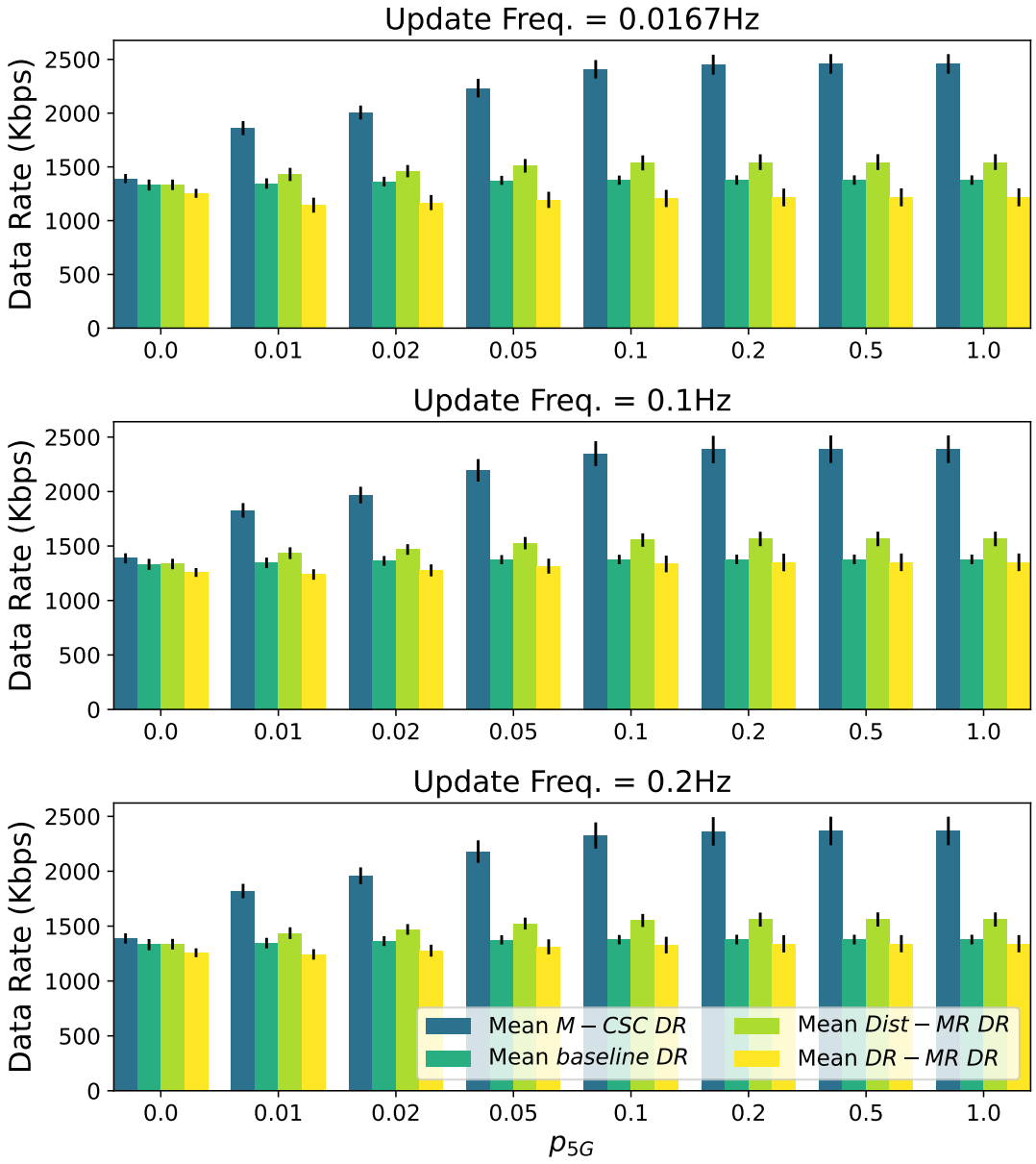

Figura 5.4: Nuovo grafico di confronto del data rate medio ottenuto con l'integrazione di SOG e COG in base alla probabilità di equipaggiamento 5G delle navi (p_{5G}), alla frequenza di esecuzione del round (f) e all'algoritmo utilizzato.

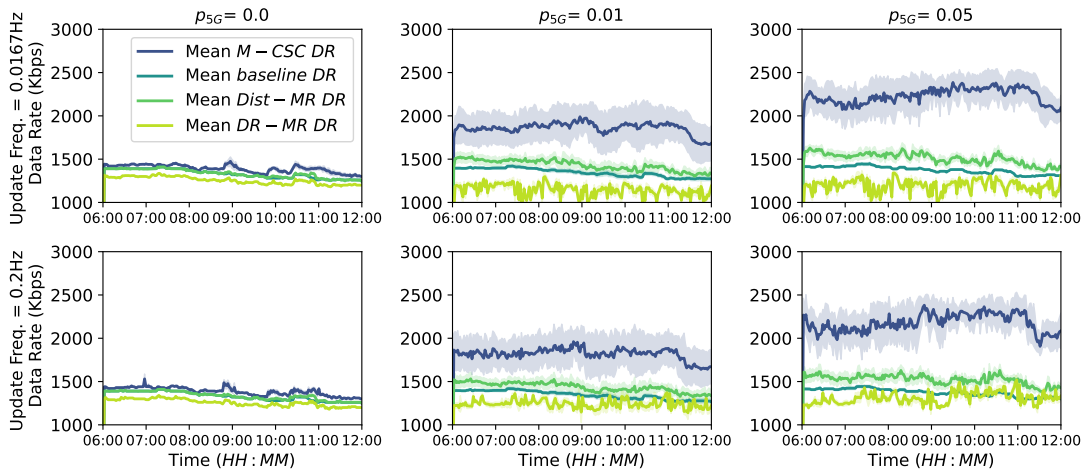

Figura 5.5: Nuovo andamento temporale del data rate medio dopo l'utilizzo di velocità e direzione delle imbarcazioni per diverse probabilità di imbarcazioni equipaggiate con torri 5G (p_{5G}), confrontando i diversi algoritmi e le diverse frequenze di aggiornamento. Le prestazioni di M-CSC aumentano anche con una limitata quantità di navi con 5G.

Riguardo all'andamento temporale dell'esperimento, in fig. 5.5 si osserva che la frequenza di aggiornamento f esercita un'influenza limitata sui risultati.

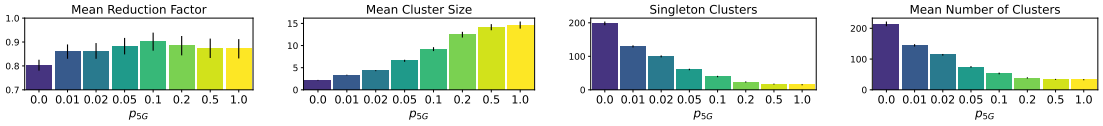

Figura 5.6: Nuove metriche per M-CSC da sinistra verso destra: fattore di riduzione medio, dimensione media del cluster, numero di cluster singoli e numero medio di cluster. Tutti i dati sono stati misurati per $f = 0,2$ Hz e per diverse probabilità di torri 5G installate sulle imbarcazioni (p_{5G}).

In conclusione, le metriche dedicate a M-CSC in fig. 5.6 mostrano i risultati ottenuti integrando SOG e COG al vecchio algoritmo. Qui, il valore massimo del fattore di riduzione si ottiene per $p_{5G} = 0.1$ e si ha un aumento della deviazione standard, probabilmente dovuto all'utilizzo di 10 seed invece che dei 100 originali. Per quanto concerne gli altri grafici, aumentando p_{5G} la media del numero di componenti nel cluster aumenta, i cluster composti da una sola imbarcazione diminuiscono notevolmente e si riduce anche il numero di cluster.

5.3 Confronto e discussione

Il confronto tra i risultati dell'esperimento precedente e quelli della nuova implementazione mostra differenze poco significative. Il data rate complessivo, riportato in fig. 5.1 e fig. 5.4, al variare della frequenza di aggiornamento e della probabilità di disponibilità di antenne 5G, mostra una riduzione di circa 100 Kbps rispetto alla versione originale, costante per ciascun algoritmo considerato. In fig. 5.7 è riportato un istogramma di confronto dei data rate medi ottenuti con il metodo CSC e con il nuovo approccio M-CSC. Nei grafici temporali (fig. 5.2 e fig. 5.5) l'andamento risulta inoltre più irregolare rispetto al caso passato. Per quanto riguarda la composizione dei cluster, il numero di cluster formati da una singola nave aumenta in corrispondenza di $p_{5G} = 0$ rispetto a prima, incrementando di conseguenza anche il numero medio di cluster nella stessa configurazione. La dimensione media dei cluster non presenta invece variazioni significative. Il fattore di riduzione medio diminuisce di circa un punto decimale per ciascuna configurazione, accompagnato da un aumento della deviazione standard.

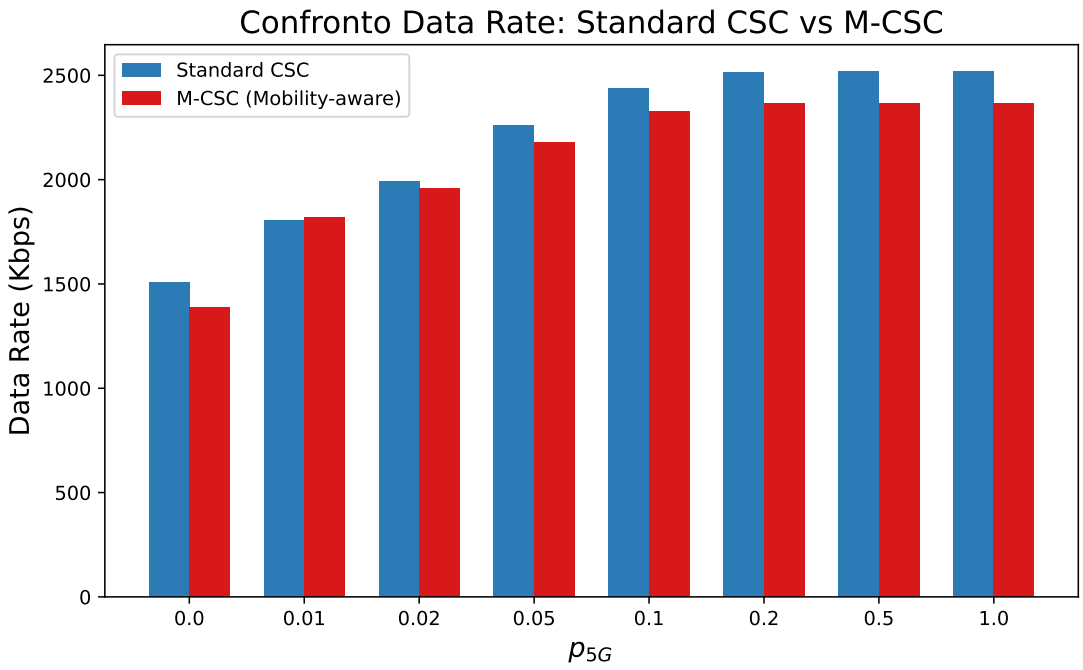

Figura 5.7: Confronto del data rate medio ottenuto dal metodo CSC standard e dal metodo M-CSC al variare della probabilità di disponibilità della rete 5G. La frequenza di aggiornamento presa in considerazione è $f = 0.2$ Hz.

Questi dati presumibilmente derivano dall'introduzione del criterio di affinità navigazionale nel processo di clustering. Probabilmente, imponendo un vincolo aggiuntivo sulla compatibilità cinematica tra le navi, il numero di vicini idonei a formare un cluster si riduce, favorendo la formazione di cluster singleton nelle configurazioni con minore densità di infrastruttura 5G. La minore aggregazione tra navi compatibili comporta a sua volta una diminuzione del fattore di riduzione medio, poiché un numero inferiore di nodi contribuisce alla sintesi delle informazioni all'interno di ciascun cluster. L'integrazione di SOG e COG non ha prodotto miglioramenti significativi nelle metriche considerate. Sebbene il criterio di affinità navigazionale abbia modificato la composizione dei cluster, l'impatto sul data rate complessivo è risultato limitato nello scenario analizzato. Questo suggerisce che la sola introduzione di informazioni cinematiche non sia sufficiente a migliorare sensibilmente le prestazioni della rete e che ulteriori strategie di integrazione della mobilità debbano essere investigate.

Capitolo 6

Conclusioni e sviluppi futuri

Questo capitolo sintetizza il lavoro svolto, riassumendo il problema di partenza, la metodologia adottata e i risultati ottenuti. Vengono inoltre discusse le limitazioni del lavoro di simulazione sviluppato e delineate le possibili direzioni per le ricerche future.

Il settore navale sta attraversando una fase di transizione verso l'impiego di navi a guida autonoma (MASS). Tali imbarcazioni necessitano di massicci flussi di dati per mantenere un'adeguata situational awareness, ma le attuali reti di comunicazione VHF sono insufficienti a supportare tali carichi [BPAFT25]. Come punto di partenza, questo lavoro ha analizzato l'algoritmo *Collective Summarisation Clusters* (CSC), sviluppato come "Fase 1" della roadmap CoMPass, il quale utilizza la programmazione aggregata per raggruppare le navi in cluster e trasmettere dati sintetizzati tramite 5G [BPAFT25, AFBPT25]. Tuttavia, basandosi esclusivamente su longitudine e latitudine delle navi, questo algoritmo ha mostrato una fragilità topologica in scenari caratterizzati da alta mobilità.

Per superare questo limite, la presente tesi ha implementato una nuova metrica consapevole della navigazione. Attraverso l'estrazione e la decodifica dei valori SOG (velocità) e COG (rotta) dai messaggi AIS, è stata formulata una funzione di penalità. Questa funzione ha permesso di modificare la scelta dei relay e l'elezione dei leader, scoraggiando la formazione di collegamenti tra imbarcazioni con traiettorie e velocità divergenti. Oltre alle modifiche apportate all'algoritmo di clustering, un contributo rilevante di questo lavoro è consistito nell'estensione del

simulatore Alchemist per supportare l'utilizzo completo dei dati AIS. In particolare, è stata progettata e implementata una pipeline che consente di decodificare i messaggi AIS, estrarre i parametri di navigazione SOG e COG, mantenerli in memoria durante l'esecuzione della simulazione ed esporli ai programmi Collektive. Questa estensione amplia le capacità del simulatore, costituendo una base riutilizzabile per futuri studi che richiedano l'impiego di ulteriori dati provenienti dai messaggi AIS.

I risultati emersi dalla simulazione hanno dimostrato che l'integrazione del criterio di affinità navigazionale influisce marginalmente sulla topologia della rete. Dal punto di vista prestazionale, il confronto con l'esperimento originario ha evidenziato che l'aggiunta delle variabili cinematiche non ha prodotto miglioramenti sul data rate complessivo, registrando anzi, in alcuni scenari, una lieve flessione (di circa 100 Kbps) e un andamento temporale più irregolare. Inoltre, in assenza di infrastruttura 5G ($p_{5G} = 0$), si è registrato un incremento dei singleton clusters (cluster composti da una singola nave) accompagnato da una diminuzione del fattore di riduzione medio. Questi esiti indicano che l'introduzione delle informazioni di rotta e velocità non è sufficiente, da sola, a innalzare sensibilmente l'efficienza della rete, suggerendo la necessità di strategie di integrazione più complesse.

6.1 Limitazioni dello studio

Nonostante l'introduzione di dati cinematici rappresenti un passo in avanti rispetto alla sola valutazione della posizione, questo studio è soggetto a diverse limitazioni. In primo luogo, l'esperimento si basa su una quantità di dati modesta, circoscritta alle traiettorie di una finestra temporale di sole sei ore all'interno del fiordo di Kiel, un campione che potrebbe non essere statisticamente rappresentativo del traffico navale su larga scala o in mare aperto [BPAFT25]. In secondo luogo, l'ambiente di simulazione risulta idealizzato. Il modello non considera la fisica reale che governa il moto delle imbarcazioni (come derive o inerzie), né l'impatto del mutare delle condizioni meteo marine (forza del mare, direzione delle onde o intensità del vento). Dal punto di vista delle telecomunicazioni, il modello assume un canale sostanzialmente affidabile basato sulla distanza, senza simulare fenomeni critici come la perdita di messaggi (*packet loss*), le collisioni sui canali, o il degrado del

segnale radio dovuto a ostacoli fisici e avversità atmosferiche. Inoltre, lo studio non tiene presente anomalie di rete, casi limite (come il guasto improvviso del nodo leader del cluster), né prende in considerazione i problemi aperti legati alla sicurezza visti nella sezione 2.1.4 [BPAFT25, BPW14]. Un'ulteriore limitazione riguarda l'impostazione statistica della simulazione. L'esperimento attuale è stato valutato utilizzando un campione di 10 seed rispetto ai 100 impiegati nella simulazione originale. Questo campione ridotto abbassa la stabilità statistica dei risultati, amplificando la deviazione standard e introducendo rumore (evidente nelle oscillazioni dei grafici temporali). Infine, i parametri introdotti per il calcolo della penalità di mobilità (come i pesi attribuiti a SOG e COG, o la soglia massima di tolleranza di 25 nodi) nel listing 4.5 sono stati determinati euristicamente. A questo si aggiunge che l'algoritmo si limita ad applicare penalizzazioni basate esclusivamente sui valori di velocità e direzione, senza estrapolare informazioni sull'evoluzione futura del sistema.

Alla luce dei risultati e delle limitazioni discusse, i futuri sviluppi di questa ricerca dovranno puntare a simulazioni più ampie e a modelli più aderenti alla realtà marittima. Dal punto di vista della valutazione, sarà necessario testare l'algoritmo su dataset AIS estesi a più giorni e su aree geografiche differenti. Inoltre, dovrà essere utilizzata la nuova funzionalità di Alchemist discussa nella sezione 4.2.4. Dal punto di vista dell'accuratezza, i simulatori dovranno integrare modelli di propagazione del segnale più complessi, che tengano conto del degrado dovuto al meteo e che introducano la perdita dei messaggi. Per superare i limiti dell'approccio odierno, SOG e COG potrebbero essere utilizzati per calcolare e prevedere la traiettoria futura delle imbarcazioni. Questo doterebbe il sistema di un comportamento di adattamento anticipativo (anticipative adaptation) [MPV12]. Reagendo in modo proattivo alle informazioni locali sugli eventi futuri (come il progressivo allontanamento di un nodo), l'infrastruttura potrebbe calcolare percorsi alternativi in modo emergente (emergent way) e riorganizzarsi prima che i collegamenti si interrompano, sfruttando i concetti dei gradienti anticipativi (*Anticipative Gradients*) [MPV12]. Infine, come indicato dalle fasi successive della roadmap CoMPass [AFBPT25], le ricerche future dovrebbero concentrarsi sull'implementazione di algoritmi di machine learning per logiche di data summarisation.

Elenco delle figure

List of Listings

Bibliografia

[AFBPT25] Ghassan Al-Falouji, Martina Baiardi, Danilo Pianini, and Sven Tomforde. Compass: A roadmap to collaborative perception and autonomy in maritime systems. In *2025 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C)*, pages 69–74, 2025.

[BPAFT25] Martina Baiardi, Danilo Pianini, Ghassan Al-Falouji, and Sven Tomforde. Robust communication through collective adaptive relay schemes for maritime vessels. In *2025 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*, pages 21–32, 2025.

[BPV15] Jacob Beal, Danilo Pianini, and Mirko Viroli. Aggregate programming for the internet of things. *Computer*, 48(9):22–30, 2015.

[BPW14] Marco Balduzzi, Alessandro Pasta, and Kyle Wilhoit. A security evaluation of ais automated identification system. In *Proceedings of the 30th Annual Computer Security Applications Conference*, ACSAC '14, page 436–445, New York, NY, USA, 2014. Association for Computing Machinery.

[BV14] Jacob Beal and Mirko Viroli. Building blocks for aggregate programming of self-organising applications. In *2014 IEEE Eighth International Conference on Self-Adaptive and Self-Organizing Systems Workshops*, pages 8–13, 2014.

[BV16] Jacob Beal and Mirko Viroli. Aggregate programming: From foundations to applications. In *Formal Methods for the Quantitative Evaluation of Collective Adaptive Systems*, pages 233–260. Springer, 2016.

[Col26] Collektive Organization. Collektive: Practical. scalable. kotlin-first aggregate programming. `https://collektive.github.io/`, 2026. Accessed: 2026.

[CPL15] Alessandro Chiurco, Leonardo Pagnotta, and Francesco Longo. *Ship bridge simulators for training in maritime domain*. PhD thesis, Scuola di Dottorato Pitagora, 2015.

[CPVN19] Roberto Casadei, Danilo Pianini, Mirko Viroli, and Antonio Natali. Self-organising coordination regions: A pattern for edge computing. In *Coordination Models and Languages: 21st IFIP WG 6.1 International Conference, COORDINATION 2019, Held as Part of the 14th International Federated Conference on Distributed Computing Techniques, DisCoTec 2019, Kongens Lyngby, Denmark, June 17–21, 2019, Proceedings*, page 182–199, Berlin, Heidelberg, 2019. Springer-Verlag.

[Dan] Danish Maritime Authority. Ais-core (dk.dma.ais). `https://github.com/dma-ais/AisLib`. Accessed: 2026.

[Int26] International Telecommunication Union (ITU). Technical characteristics for an automatic identification system using time division multiple access in the maritime mobile service. Recommendation Recommendation ITU-R M.1371-6, ITU Radiocommunication Sector, Geneva, Switzerland, 02 2026.

[MPV12] Sara Montagna, Danilo Pianini, and Mirko Viroli. Gradient-based self-organisation patterns of anticipative adaptation. In *2012 IEEE Sixth International Conference on Self-Adaptive and Self-Organizing Systems*, pages 169–174, 2012.

[Pia21] Danilo Pianini. Simulation of Large Scale Computational Ecosystems with Alchemist: A Tutorial. In Miguel Matos and Fabíola Greve,

editors, *Lecture Notes in Computer Science*, volume LNCS-12718 of *Distributed Applications and Interoperable Systems*, pages 145–161, Valletta, Malta, June 2021. Springer International Publishing. Part 5: Invited Paper.

[PMV13] D Pianini, S Montagna, and M Viroli. Chemical-oriented simulation of computational systems with alchemist. *Journal of Simulation*, 7(3):202–215, August 2013.

[Uni15] United States Coast Guard. 2015 code of federal regulations implementing the ais provisions of the safety of life at sea (solas) convention and marine transportation security act of 2002, 2015. Includes IMO Resolution MSC.99(73) and U.S. Code Title 46.

[VAB+18] Mirko Viroli, Giorgio Audrito, Jacob Beal, Ferruccio Damiani, and Danilo Pianini. Engineering resilient collective adaptive systems by self-stabilisation. *ACM Trans. Model. Comput. Simul.*, 28(2), March 2018.

Acknowledgements

Ringrazio la mia famiglia, Terry, Walther, Enri, Sofi e Ale per l'aiuto costante e l'amore incondizionato che mi date ogni giorno. Grazie alla mia famiglia slovacca, Lolo, Iveta, Elena e Petra per avermi accolto come se fossi vostro figlio. Grazie ai miei compagni di università, Tommi, Giuly, Ele, Carmine, Fabio, Dave per avermi accompagnato in questo viaggio e al mio bro-getto Luca. Ringrazio i miei amici Vito e Cate per sostenermi sempre e per le nostre serate Trivial; grazie a Filip per esserci da sempre. Infine, il ringraziamento più importante va alla mia metà e all'amore della mia vita, Vichi, senza la quale non avrei nemmeno immaginato di potermi diplomare. Sei il motivo per cui mi impegno e vado avanti ogni giorno, per poter essere il compagno di vita che meriti e per creare il miglior futuro possibile con te.