POLITECNICO DI TORINO

Corso di Laurea Magistrale in Ingegneria Informatica

Tesi di Laurea Magistrale

Routing Dinamico nel Physical Internet:

dalla Simulazione al Mock

Politecnico di Torino

Relatore:
Prof. Giovanni Zenezini

Correlatore:
Prof. Massimo Rebuglio

Candidata
Petra Mirtilla Cecchi

Novembre 2023

Sommario

Il Physical Internet (PI) è un sistema innovativo che ha lo scopo di rivoluzionare gli attuali paradigmi della logistica globale. Si fonda sui principi del Digital Internet, come la trasparenza, la condivisione delle risorse e lo scambio di informazioni, con l'obiettivo di creare una rete logistica più efficiente ed attenta alle tematiche ambientali. Sono stati proposti diversi protocolli di routing per i PI-container, le unità di carico che vengono proposte nel PI, e questi protocolli coinvolgono anche l'interazione con i nodi della rete, i PI-hubs. Allo stato dell'arte, analizzando questi protocolli emergono due principali limitazioni: innanzitutto, contengono componenti non reali; inoltre, sono difficili da interfacciare con altre simulazioni o componenti fisici. Questa tesi propone una simulazione di routing "multilanes" (next-hop) che è stata debitamente adattata per essere facilmente interfacciata con sistemi reali, rendendola utilizzabile come Mock.

Indice

Elenco delle figure

Elenco dei listati

Capitolo 1

Introduzione

1.1 Physical Internet

Il concetto di Physical Internet nasce nel 2006 dalla copertina di un articolo dell'Economist [4] che ha come scopo l'analisi dello stato dell'arte della logistica.

Figura 1.1: Copertina dell'Economist - 17/06/2006

A partire da questa immagine, Benoit Montreuil[1] (attualmente professore presso il Georgia Institute of Technology) inizia a teorizzare una nuova architettura per

poter rendere realtà l'idea di un vero e proprio Internet fisico, dove al posto di pacchetti virtuali vengono trattate merci fisiche che devono muoversi al di sopra di una rete (stradale, aerea, navale,..) per raggiungere una destinazione finale. In questo capitolo verranno analizzati i concetti principali che sono alla base del Physical Internet, gli attori che ne fanno parte, le analogie e le differenze con il Digital Internet e infine i vantaggi ipotizzati dai suoi ideatori rispetto alle architetture logistiche classiche. I risultati di questi primi studi sono stati pubblicati qualche anno dopo nel libro "The Physical Internet: The Network of logistics Networks"[5]. Inoltre, è disponibile online il manifesto[11] di questa teoria che riassume gli aspetti teorici fondamentali di riferimento sulla modalità di implementazione del PI e anche dei benefici di un'architettura di questo tipo rispetto alla logistica classica alla base della trattazione di questa tesi.

1.1.1 Il concetto di Physical Internet

L'idea principale dietro al concetto del PI è molto semplice: nello stesso modo in cui nella rete Internet si è riusciti attraverso l'utilizzo di protocolli noti e comuni ad interconnettere reti di calcolatori basate su tecnologie completamente diverse, nel PI si vogliono interconnettere reti logistiche utilizzando dei protocolli standard di collaborazione, la possibilità di dividere la merce in container modulari e interfacce intelligenti tra i vari attori per incrementare efficienza e sostenibilità[5]. Il primo aspetto da analizzare è quello dell'interconnessione tra le diverse reti di logistica esistenti, quindi non vedere più ogni azienda di trasporti o di spedizioni come un'isola separata dal resto, ma come attori di una collaborazione per poter ottimizzare il transito delle merci dalla partenza alla destinazione. Per questo motivo nella sua fase iniziale ci si rifà al DI per poter prendere spunto dai suoi protocolli che sono serviti proprio per standardizzare il passaggio dei dati (in questo caso delle merci) da una rete ad un'altra. Il secondo aspetto è quello relativo ai container modulari. Questa idea prende spunto dalle spedizioni via nave che prevedono i container per ottimizzare lo spazio. Dal punto di vista del PI, i container giocano un ruolo fondamentale per poter ottimizzare lo spazio disponibile su ogni tipologia di mezzo ed accorpare il più possibile le merci che devono raggiungere lo stesso luogo. L'ultimo aspetto riguarda le interfacce tra gli attori di questa architettura, in qualche modo bisogna prevedere una o più figure intermediarie che coordinino il passaggio delle merci tra un'azienda e l'altra, oppure che si occupino di assegnare la merce ai container in modo appropriato, il successivo tracciamento della merce e che si interfaccino con gli utenti del sistema logistico, chi spedisce e chi riceve la merce.

1.1.2 Differenze tra logistica classica e PI

La logistica classica ha per lungo tempo operato in maniera segmentata, con ogni azienda che gestisce autonomamente la propria catena di fornitura. Questo approccio, sebbene familiare, porta spesso a inefficienze, soprattutto quando si considera l'ottimizzazione del trasporto e la sottoutilizzazione dei mezzi. Ad esempio, non è raro vedere camion o navi semi-vuoti che percorrono lunghe distanze, contribuendo a emissioni di carbonio inutili e costi elevati. Inoltre, la mancanza di standardizzazione nelle pratiche logistiche rende difficile l'interoperabilità tra le diverse aziende, complicando ulteriormente il quadro.

Il concetto di PI si presenta come una rivoluzionaria alternativa a questa situazione. Ispirandosi ai protocolli e alle strutture del Digital Internet, il PI propone un approccio integrato e collaborativo alla logistica. Le aziende non sono più viste come entità isolate, ma come nodi interconnessi all'interno di una rete logistica globale. Questo permette una condivisione delle risorse e delle informazioni, con l'obiettivo di ottimizzare la catena di fornitura su scala globale. Uno degli aspetti chiave del PI è l'uso di container modulari, che permettono di massimizzare l'efficienza del trasporto e ridurre le emissioni. Inoltre, la standardizzazione e l'adozione di protocolli comuni facilitano la comunicazione e la collaborazione tra aziende, garantendo una maggiore tracciabilità delle spedizioni. Nel complesso, mentre la logistica classica può sembrare più familiare e consolidata, il PI offre un quadro di efficienza, sostenibilità e collaborazione che potrebbe rivoluzionare il modo in cui vediamo la logistica nel futuro.

1.1.3 Organizzazione del Physical Internet

Nella visione del PI, la logistica non è solo una serie di operazioni isolate, ma un ecosistema interconnesso che funziona in modo simile alla rete di computer del Digital Internet. Al centro di questa visione c'è l'idea di una struttura aperta, flessibile e collaborativa, che vada oltre le convenzionali barriere aziendali e settoriali.

Interconnessione e protocolli: Allo stesso modo in cui Internet si basa su protocolli standardizzati per facilitare la comunicazione tra computer diversi, il PI adotta protocolli comuni per garantire l'interoperabilità tra diversi sistemi logistici. Questi protocolli consentono la condivisione efficiente delle risorse, come i mezzi di trasporto o i magazzini, e facilitano l'ottimizzazione della catena di fornitura a livello globale.

Modularità: Un principio chiave del PI è l'uso di container modulari standardizzati. Questi container, analogamente ai pacchetti di dati nel Digital Internet, sono progettati per essere trasportati facilmente su diversi mezzi di trasporto (camion, treni, navi, ecc.), rendendo il sistema più flessibile e resiliente. L'obiettivo è massimizzare l'efficienza nello spostamento delle merci, riducendo gli spazi vuoti e ottimizzando i percorsi.

Hub e nodi: Analogamente ai nodi di routing nel Digital Internet, il PI ha hub logistici chiave che fungono da punti di transito principali per le merci. Questi hub sono strategicamente posizionati per garantire la distribuzione efficiente delle merci e sono dotati di tecnologie avanzate per la gestione, il tracciamento e l'ottimizzazione delle spedizioni.

Tecnologia e automazione: Il PI è profondamente integrato con tecnologie avanzate come l'Internet delle Cose (IoT), l'intelligenza artificiale e la blockchain. Queste tecnologie forniscono una tracciabilità in tempo reale, automazione e soluzioni predittive che rendono il sistema più efficiente e affidabile.

In conclusione, l'organizzazione del Physical Internet rappresenta una svolta radicale rispetto ai tradizionali sistemi logistici, puntando a un'efficienza, sostenibilità e collaborazione senza precedenti. Facilitando una visione condivisa e integrata della logistica, il PI ha il potenziale di rivoluzionare il modo in cui le merci vengono trasportate e distribuite a livello globale.

1.1.4 Vantaggi del Physical Internet

Nel panorama della logistica moderna, il PI emerge come una soluzione rivoluzionaria, portando con sé una serie di vantaggi distintivi. Al primo posto, troviamo l'efficienza operativa. L'adozione di protocolli comuni e l'interconnettività tra le aziende permettono una condivisione di risorse, come mezzi di trasporto e magazzini. Questa collaborazione riduce la sottoutilizzazione dei mezzi, contribuendo a minimizzare i costi operativi.

La sostenibilità ambientale è un altro grande vantaggio del PI. In un mondo sempre più attento alle tematiche ambientali, l'ottimizzazione dei trasporti e la riduzione dei percorsi inutili o semi-vuoti rappresentano una significativa diminuzione delle emissioni di carbonio. Questo non solo ha un impatto positivo sull'ambiente, ma porta anche a considerevoli risparmi economici per le aziende, in termini di carburante e manutenzione.

Un ulteriore beneficio del PI è la trasparenza e la tracciabilità. Grazie alla tecnologia e all'adozione di standard comuni, le aziende hanno la possibilità di monitorare le merci in tempo reale. Questo offre una maggiore sicurezza e affidabilità nel processo di spedizione, migliorando la fiducia tra gli attori coinvolti e riducendo i tempi di consegna.

In sintesi, l'adozione del PI può portare a un'evoluzione significativa nella logistica, offrendo soluzioni più efficienti, sostenibili e trasparenti.

1.2 Il routing nel Physical Internet

Nel PI, il routing delle merci o meglio dei PI-containers è uno degli aspetti fondamentali da tenere in considerazione per garantire l'efficienza della gestione delle merci sia in termini di scelta del percorso (minor tempo di trasporto), ma anche di

emissioni (ad esempio con la scelta di riempire completamente un unico container piuttosto che più container separati). In questa sezione, verranno analizzate similitudini e differenze tra il routing nel PI rispetto agli approcci classici di routing nel Digital Internet e verrà fatta una panoramica degli algoritmi di routing allo stato dell'arte nel PI.

1.2.1 Modello ISO/OSI, OLI e NOLI

Per capire meglio di cosa si deve occupare il routing nella PI, bisogna andare a vedere il parallelismo individuato tra il modello ISO/OSI (Open Systems Interconnection) in utilizzo per l'astrazione delle funzionalità e dei livelli di rete nel DI e i modelli OLI (Open Logistics Interconnection) e NOLI (Open Logistics Interconnection) presentati per gestire il parallelismo con il DI e prenderne spunto per dividere in livello con protocolli e funzionalità ben definite anche il PI.

Il modello ISO/OSI è lo standard concettuale sviluppato per descrivere come diversi protocolli di rete interagiscono e comunicano tra loro. Il modello è suddiviso in sette strati, o livelli, ciascuno dei quali svolge una funzione specifica nel processo di comunicazione e permette di assegnare i vari protocolli standard per ogni livello. La separazione di funzionalità consente di facilitare lo sviluppo, la manutenzione e l'interoperabilità dei protocolli di rete e di costruire protocolli di alto livello che si appoggiano su tutti quelli sottostanti. Questo modello è stato un contributo significativo nello sviluppo delle reti di computer e fornisce un quadro concettuale ancora utilizzato oggi, ed è stato alla base di modelli derivati da lui come ad esempio il TCP/IP attualmente in utilizzo nelle reti di calcolatori.

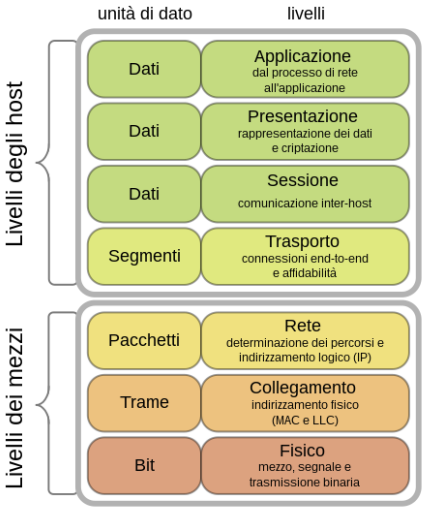

Figura 1.2: Modello ISO/OSI[7]

A partire dal modello OSI per il DI, sono stati presentati due modelli applicabili nel PI.

Il modello OLI[9] è il primo modello presentato e consiste nell'individuazione di 7 livelli applicabili al PI.

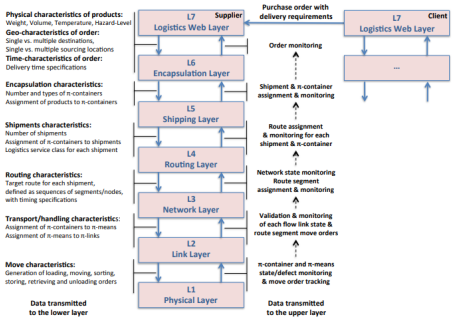

Figura 1.3: Modello OLI[9]

Per ogni livello, come si può vedere nell'immagine, sono state individuate anche le funzioni che ogni livello deve fornire a quello successivo. Si può vedere che sono stati individuati un network layer e un routing layer che insieme si occupano della gestione dello spostamento delle merci da un PI-hub ad un altro. Il primo si occupa della gestione più di basso livello relativa all'assegnamento dei carichi piuttosto che la composizione/scomposizione dei containers una volta arrivati a destinazione (la funzione svolta nel DI dal TCP), mentre il livello di routing si occupa dell'assegnazione delle rotte ad ogni container in modo da garantirne il trasporto dall'origine alla destinazione.

L'ultimo modello proposto invece è il NOLI[3] che propone una suddivisione dei livelli basata sull'OLI, ma con più focus al risvolto pratico di ogni strato.

Come si può vedere dalla tabella, diversamente dal modello precedente, il livello di trasporto (Transport Layer) riceve origine e destinazione dei container, il livello di rete (Network Layer) si occupa di definire il percorso che il container dovrà seguire ed infine il livello di collegamento (Link Layer) si occupa dello spostamento tra un PI-hub ed un altro gestendo così i vari hop per raggiungere la destinazione finale.

In conclusione, i modelli discussi forniscono un punto di riferimento essenziale non solo per adattare i concetti fondamentali della rete DI alla rete PI, ma anche per identificare in modo esaustivo i protocolli e le funzionalità indispensabili per la realizzazione della PI. Tali modelli fungono anche da guida per l'implementazione delle diverse funzionalità. Questo tenendo conto sia dei processi automatizzati, come ad esempio l'utilizzo di algoritmi di routing ad-hoc per determinare il percorso ottimale tra punto di partenza e destinazione, sia dei processi nei quali sarà necessario l'intervento umano.

Position in the NOLI model	Layer Name	Role of the Layer
7	Product Layer	Defines the possible products or goods that can be transported inside π-containers. It fills the π-containers with the products and establishes the related contracts.
6	Container Layer	Defines the physical characteristics of the π-containers allowed on the Logistics Network. It will check the physical integrity of the π-containers and combine them into "sets" according to their characteristics.
5	Order Layer	Receives sets of π-containers from the Container Layer. It will create the orders according to the specified constraints (deadlines, client whishes, starting and destination point, etc.), and assigns the π-containers to the orders.
4	Transport Layer	Receives orders made of π-containers from the Order Layer. The transport Layer creates "loads" from the received orders, and manages the end-to-end trip for each load.
3	Network Layer	Receives loads of π-containers from the Transport Layer and creates "blocks" from the loads. The Network Layer defines a path across the network for each block.
2	Link Layer	Manages the individual steps (point-to-point movement) of π-containers on π-means.
1	Physical Handling Layer	Physical characteristics description of the π-means used to move the containers.

Figura 1.4: Modello NOLI[3]

1.2.2 Parallelismo con il routing nel DI

In questa sezione, si andranno ad analizzare, solamente per quanto riguarda il processo di routing, le somiglianze e le differenze tra i classici modelli di routing nel DI rispetto agli algoritmi di routing che possono essere applicati nel PI.

Innanzitutto, bisogna tener conto della diversa natura della merce trasportata: se nel DI troviamo pacchetti virtuali trasmessi tramite onde elettromagnetiche piuttosto che ottiche, nel PI bisogna tener conto che si ha sempre a che fare, in ogni livello, con elementi fisici. Sel nel DI spesso ci si può accontentare di algoritmi best-effort, semplicemente perché in caso di perdita di pacchetti questi possono essere ritrasmessi, nel caso del PI le merci da spedire potrebbero avere una serie di vincoli da rispettare ad esempio tempistiche certe per la consegna, piuttosto che vincoli ambientali (deperibili, trasporti eccezionali, infiammabili). Perciò, gli algoritmi di routing nel PI devono essere pensati in modo dinamico, per adattarsi alle possibili modifiche del percorso, perché le strade da percorrere possono essere interrotte o possono esserci dei guasti.

Inoltre, il vero trasporto viene effettuato da autisti in carne ed ossa e non da router e switch che possono operare 24 ore su 24, senza interruzioni, quindi anche la componente umana gioca un ruolo importante sulle scelte algoritmiche perché bisogna considerare la necessità dell'autista di fermarsi nel tragitto o trovare ingorghi e non sempre i tempi di percorrenza calcolati possono essere garantiti.

Infine, negli algoritmi di routing del DI il peso dei collegamenti viene calcolato in termini di banda oppure latenza, alcuni collegamenti sono preferibili ad altri

per minimizzare i tempi di trasmissione. Nel caso della PI, non è certo che per tutti gli attori in gioco il tempo minimo di percorrenza sia la metrica principale per effettuare le scelte. Si potrebbero voler ottimizzare i carichi dei container per evitare mezzi vuoti in transito oppure utilizzare il più possibile mezzi a basse emissioni piuttosto di quelli a carburanti fossili.

In conclusione, gli algoritmi di routing nel DI devono essere pensati in modo da poter essere efficaci anche in caso di metriche non per forza legate al cammino minimo sul tragitto partenza-destinazione.

1.3 Simulazioni a tempo discreto

Le simulazioni a tempo discreto rappresentano uno strumento fondamentale nella teoria dei sistemi e nel campo dell'ingegneria delle informazioni. Questo tipo di simulazione tratta il tempo come una serie di punti distinti o intervalli, piuttosto che come una continua variabile fluida, come si vedrebbe nelle simulazioni a tempo continuo. In pratica, ciò significa che il sistema viene analizzato in momenti specifici, piuttosto che in ogni istante infinitesimale. Ogni volta che un evento avviene, il sistema aggiorna il suo stato in base a tale evento e procede al successivo, ignorando gli intervalli di tempo in cui non si verificano cambiamenti.

Il vantaggio di questo approccio è che consente di concentrarsi sulle modifiche rilevanti per il sistema, piuttosto che sul monitoraggio continuo. Ciò può semplificare notevolmente l'analisi, in particolare in sistemi complessi in cui solo pochi eventi sono cruciali per la dinamica generale del sistema. Questa caratteristica rende le simulazioni a tempo discreto particolarmente adatte per settori come la logistica, in cui gli eventi chiave, come la spedizione o la ricezione di merci, sono spesso separati da periodi di inattività o attività di routine.

Nel contesto del PI, le simulazioni a tempo discreto possono giocare un ruolo cruciale. Il PI, essendo una rete globale e complessa di flussi di merci, presenta innumerevoli variabili e interazioni. Immaginiamo, ad esempio, un hub logistico all'interno del PI. Gli eventi chiave in questo hub potrebbero includere l'arrivo di un camion, il caricamento o lo scarico di merci, o il trasferimento di merci da un magazzino all'altro. Invece di modellare ogni singolo secondo di attività in questo hub, una simulazione a tempo discreto potrebbe concentrarsi solo sui momenti in cui questi eventi avvengono, rendendo la simulazione molto più gestibile e informativa.

Inoltre, le simulazioni a tempo discreto permettono di anticipare e prevedere le potenziali congestioni, i ritardi e le inefficienze all'interno del sistema logistico. Per esempio, se una simulazione predice che due grandi spedizioni arriveranno nello stesso hub logistico allo stesso tempo, potrebbero essere necessari aggiustamenti per prevenire ingorghi o ritardi. Questo tipo di previsione proattiva è fondamentale per garantire che il PI funzioni in modo ottimale.

In sintesi, le simulazioni a tempo discreto offrono uno strumento potente e flessibile per modellare e ottimizzare sistemi complessi come il Physical Internet. Attraverso la loro capacità di focalizzarsi sugli eventi chiave e di ignorare i periodi di inattività, queste simulazioni possono fornire intuizioni preziose sulla dinamica dei sistemi logistici e aiutare a guidare decisioni informate per migliorare l'efficienza e la reattività del PI.

Capitolo 2

Stato dell'arte

In questo capitolo verrà analizzato lo stato dell'arte al momento della pubblicazione del paper "A dynamic routing protocol with payments for the Physical Internet: A simulation with learning agents"[2] preso in esame relativo al primo tentativo conosciuto di creare un algoritmo funzionante per poter simulare un'implementazione reale di routing nel PI utilizzando un sistema ad aste. Queste aste sono pensate per essere eseguite in tempo reale nei nodi (che rappresentano i PI-hubs) che ne coordinano dati, assegnazione e conseguente prossimo nodo di destinazione. L'algoritmo proposto utilizza alcuni concetti propri dello smistamento pacchetti del Digital Internet per poter scegliere il migliore next hop e di conseguenza il miglior trasportatore al quale affidare la spedizione. La pubblicazione presenta sia l'algoritmo per l'assegnazione delle aste che l'ambiente utilizzato per la simulazione di utilizzo nell'ambito della PI, basato su procedure di autoapprendimento dei nodi e dei trasportatori per poter generare un traffico realistico tra i nodi ed una partecipazione imparziale alle aste. Inoltre, il tutto viene confrontato con i risultati di una simulazione con gli stessi nodi e gli stessi trasportatori, ma in una situazione di routing nella PI non basata sul next hop, ma sulla single-lane, ovvero l'assegnazione di un carico ad un trasportatore dal nodo di partenza a quello di destinazione.

2.1 Scenario di routing

In questa sezione verranno presentati gli attori necessari per l'esecuzione dell'algoritmo di routing presentato e il loro comportamento. Lo scenario prevede il modello dei principali componenti teorizzati nel PI, in alcuni casi semplificati a scopo della simulazione.

2.1.1 I nodi - Nodes

I nodi (o Nodes) possono essere visti come i luoghi fisici dove vengono centralizzate e smistate le merci (warehouse, centri di smistamento, magazzini, ecc) dove i vari

trasportatori consegnano le merci trasportate e attendono di ricevere della nuova merce da trasportare verso un altro nodo. Dal punto di vista logico-algoritmico, svolgono i seguenti compiti:

Operazioni sulle merci: il nodo mantiene la lista ordinata delle merci in attesa di essere spedite e il loro nodo di destinazione. Si possono avere due tipologie di merci, quelle che hanno come punto di partenza il nodo stesso oppure possono essere merci in transito, ma non ancora arrivate a destinazione. In questo scenario, questi due casi vengono trattati esattamente nello stesso modo dato che in ogni nodo viene contrattato il trasporto solo per il nodo successivo (o next-hop) che non necessariamente sarà il nodo della destinazione definitiva.

Operazioni sui trasportatori: il nodo mantiene la lista dei trasportatori in attesa di partecipare ad un'asta ed essere scelti, se vincenti, per trasportare la merce. Inoltre, per ogni asta, decide quali tra questi trasportatori può partecipare con lo scopo di mantenere un equilibrio nella competizione ed abbattere i tempi di attesa nell'esecuzione delle aste.

Svolgimento delle aste: ogni volta che almeno un trasportatore e almeno un carico sono in attesa di essere assegnati e spediti, svolgono un'asta per creare la coppia trasportatore-merce fino a quando le merci in attesa non vengono spedite tutte.

Gestione dei pagamenti: dato che assumono anche il ruolo di intermediari, nel caso di questo scenario i nodi gestiscono anche i pagamenti che possono essere di due tipi. Quando un trasportatore viene scelto per trasportare della merce, il nodo chiede a chi spedisce di pagarlo. Inoltre, in alcuni casi l'intermediazione e la logistica che prevede un costo fisso e quindi il nodo chiede a chi spedisce di pagarlo per i servizi svolti.

Essendo in uno scenario di PI, diversamente da quello che accade nella logistica classica, i nodi non appartengono né a chi effettua le spedizioni né alle aziende trasportatrici, ma sono da immaginare come dei luoghi neutrali nei quali convergono tutte le merci, in transito e da spedire, e che fungono da intermediari tra chi spedisce e chi trasporta, talvolta prevedendo delle spese tecniche per la gestione dell'intermediazione.

2.1.2 I trasportatori - Carriers

I trasportatori (o Carriers) sono fisicamente i mezzi che trasportano la merce da un nodo ad un altro. In generale, ogni azienda di logistica dovrebbe possedere una serie di trasportatori che possono essere immaginati come camion, corrieri, ma anche containers per le spedizioni navali e ferroviarie. In questo caso, modellano anche le persone fisiche che in ogni nodo attendono di vincere un'asta, ricevere la merce e la destinazione e compiere lo spostamento. Dal punto di vista logico-algoritmico, svolgono i seguenti compiti:

Partecipazione alle aste: ogni volta che si trovano fermi ad un nodo, possono partecipare, se scelti dal nodo, ad una o più aste. Nel caso in cui si aggiudichino l'asta, ricevono il pagamento e partono per il nodo di destinazione previsto secondo la strada decisa. Nel caso in cui non si aggiudichino l'asta, possono partecipare a quelle successive. Nel caso le perdessero tutte possono scegliere se rimanere in attesa, spostarsi vuoti in un altro nodo oppure tornare a casa.

Spostamento: lo spostamento avviene tra due nodi direttamente collegati senza passare per nodi intermedi, quindi può essere visto come un next-hop dato che l'algoritmo presentato frammenta il percorso completo della merce in tanti hop che teoricamente dovrebbero essere il cammino minimo tra nodo di partenza e nodo di destinazione. In realtà, dato che vengono gestiti dalle aste, non è detto che questo sia il cammino minimo dal punto di vista chilometrico, ma sicuramente è il cammino minimo dal punto di vista dei costi complessivi.

2.1.3 I mittenti - Shippers

I mittenti (o Shippers) della merce sono le compagnie che inviano i beni verso i clienti business o privati, ovvero coloro che generano i carichi che vanno trasferiti tra un nodo di partenza e uno di destinazione. Possono essere visti come i vari mittenti della merce spedita online oppure direttamente da un fornitore al ricevente. Dal punto di vista del mercato reale possono essere visti come i reali attori del mercato virtuale (Amazon, Ebay, Subito, etc) oppure aziende singole o addirittura privati. Al giorno d'oggi, chiunque può essere visto come mittente, dato che chiunque può generare merce che viene spedita tramite corrieri o mezzi di trasporto terzi. Dal punto di vista logico-algoritmico, svolgono i seguenti compiti:

Generazione delle merci: in ogni momento, i mittenti notificano in ogni nodo la presenza di merci da spedire. Oltre alla destinazione, comunicano anche un reserve price che è da vedere come il tetto massimo che sono disposti a pagare per ogni asta.

Partecipazione alle aste: ogni volta che una merce spedita da un certo mittente si trova ferma in un nodo, il mittente comunica il reserve price per il nodo successivo e permette così al nodo di effettuare l'asta. Nel caso nessun trasportatore offra un prezzo inferiore al reserve price può scegliere se alzarlo per effettuare un'altra asta oppure di cancellare la spedizione.

Pagamenti: come già preannunciato, il mittente è colui che paga per la spedizione il trasportatore e per il servizio il nodo. I costi del trasportatore sono decisi tramite asta, mentre quelli del nodo sono noti a priori.

2.1.4 I collegamenti - Lanes

I collegamenti (o Lanes) tra un nodo e l'altro sono come gli archi in una rete informatica. Nel caso del PI, sono effettivamente le strade oppure le linee di collegamento

(aeree, navali, ferroviarie) tra un nodo e l'altro. Ogni arco ha un peso, in generale scelto in base alla distanza chilometrica da percorrere. Per questo, la scelta della strada da parte dei trasportatori gioca un ruolo molto importante nelle aste e nel risultante percorso complessivo della merce. Non sono un attore attivo nel modello di questo algoritmo, ma le loro caratteristiche giocano un ruolo importante in quanto fisicamente sono il mezzo reale sul quale si muovono le merci e quindi sono tenute in conto quando ogni trasportatore va a proporre un prezzo per ogni strada percorribile.

2.2 Le aste

In questa sezione verrà presentato l'algoritmo relativo alle modalità di esecuzione delle aste e all'interazione tra gli attori presentati nella sezione precedente. Le aste sono il cuore logico della soluzione presentata in quanto rappresentano la modalità di scelta del next-hop nell'algoritmo di routing presentato.

2.2.1 Situazione iniziale

Come detto precedentemente, ogni nodo esegue un'asta o più aste nel momento in cui sono presenti almeno un carico da spedire ed un trasportatore fermi presso di lui. Per far questo, ogni nodo mantiene una lista di merci in attesa e trasportatori in attesa, dato che nell'esecuzione generale le tempistiche delle operazioni vengono gestite a tempo discreto. Ogni nodo è un sistema a sé e gestisce le proprie aste. Ogni nodo conosce quindi un set di collegamenti (o Lanes) ad esso adiacenti, un set di trasportatori (o Carriers) in attesa di partecipare alle aste e un set di merci in attesa di essere spedite. In più, per ognuna di queste merci, conosce un reserve price, generato dal mittente (o Shipper), che tiene conto del prezzo massimo che quest'ultimo può permettersi di pagare per quell'asta.

2.2.2 Meccanismo di asta

La sequenza delle operazioni che compongono un'asta è la seguente:

Merce e reserve price: il mittente comunica per la merce presa in esame il reserve price e lo comunica al nodo. Questo reserve price tiene con del nodo di partenza e quello di destinazione e deve essere abbastanza alto da permettere ai trasportatori di puntare nell'asta un prezzo che possa essere competitivo, ma che non sia più basso dei reali costi del trasporto merce più eventuali costi fissi.

Generazione dei pesi: a questo punto il nodo genera dei pesi per ogni collegamento ad ogni nodo adiacente. In generale questi pesi devono essere direttamente proporzionali alla distanza tra un nodo e l'altro percorrendo quel collegamento. In pratica, nel caso reale questi pesi potrebbero dipendere anche da altri fattori (percorribilità della strada, eventuali limitazioni, traffico, interruzioni,...). Nel caso

dell'algoritmo, il nodo cerca di fornire per ogni peso la migliore stima che possa fare rispetto al costo che ci sarà dal nodo successivo alla destinazione.

Esecuzione dell'asta: dopo aver generato i pesi, il nodo bandisce un'asta per la prima merce presente in lista e chiede ai trasportatori selezionati di partecipare generando un prezzo per ogni strada verso i nodi adiacenti. Il prezzo generato dai trasportatori è un valore finito per ogni strada percorribile, mentre propone un valore infinito per le strade non percorribili.

Output dell'asta: dopo aver ricevuto le proposte di ogni trasportatore disponibile per ogni collegamento, il nodo proclama il vincitore dell'asta comunicandoglielo e notificando il pagamento da effettuare al mittente. Nel caso in cui il vincitore scelto abbia proposto un prezzo maggiore del reserve price, l'asta viene cancellata e si chiede al mittente di proporre un nuovo reserve price oppure di ritirare la merce.

2.3 La simulazione

In questa sezione verrà presa in esame la simulazione completa condotta per testare l'algoritmo di routing tramite aste presentato nella sezione precedente. Innanzitutto va detto che la simulazione è divisa in due componenti principali: un ambiente che funge da orologio per svolgere le varie azioni e che presenta una simulazione a tempo discreto per poter gestire gli spostamenti dei trasportatori da un nodo ad un altro e una preparazione preliminare di questo ambiente per poter creare una situazione realistica per poter confrontare questo algoritmo di routing con un algoritmo classico per il trasporto delle merci. Già da questi presupposti, si può intuire che i test svolti, sebbene basati in alcuni casi su dati reali, sono completamente artificiali e non prevedono nessuna interazione con il mondo reale. In ogni caso, i dati generati da questa simulazione indicano che l'algoritmo in questione può essere considerato valido quando confrontato con gli approcci di routing attualmente impiegati nella logistica tradizionale. I risultati ottenuti forniscono una prospettiva positiva sulla sua efficacia e suggeriscono che potrebbe rappresentare un'alternativa vantaggiosa rispetto alle metodologie consolidate.

2.3.1 Vincoli

Il primo argomento da affrontare è relativo ai vincoli proposti per la semplificazione dell'ambiente di simulazione. Sia dal punto di vista degli attori presenti che dal punto di vista dei dati di input sono state eseguite delle semplificazioni per poter ingegnerizzare in tempo breve l'algoritmo. Di seguito, vengono elencati i presupposti essenziali che sono stati presi in considerazione dagli autori dell'algoritmo.

Full Truck Load: diversamente da come ipotizzato spesso nel PI, ogni trasportatore viene modellato come un unico container che può contenere al massimo un carico. In realtà, questa è una limitazione rispetto ad uno dei concetti fondamentali

del PI, in cui si vanno ad ipotizzare dei container che possono contenere varie tipologie di merci composte da colli di tipo diverso e con diversi mittenti. Sicuramente, questa semplificazione aiuta ai fini dell'algoritmo perché ad ogni asta corrisponde un unico carico che va a riempire un unico camion.

Nodi: i nodi presi in considerazione sono prefissati. Sono 11 città dell'Europa occidentale che sono effettivamente dei punti di interscambio per la logistica. In questo modo il grafo composto da questi nodi è predefinito sia per quanto riguarda la posizione uno rispetto all'altro che rispetto alle effettive distanze, utili ai fini di calcolare i pesi dei vari collegamenti. Inoltre, tutti i dati relativi al numero di carichi generati in ogni nodo e alla probabilità di un carico con un certo nodo di partenza e nodo di destinazione sono modellati a partire da dati reali presi dal database ETIS+ che permette di aggiungere realismo alla simulazione.

Mittenti: nonostante normalmente i carichi siano generati da vari mittenti, nel caso di questo caso di studio in tutta la simulazione tutti i carichi sono generati da un unico mittente. Questa limitazione non è così complicata da aggirare dato che semplicemente lo studio risulta come un caso specifico di scenario che potrebbe essere esteso a più mittenti eterogenei.

Scelte strategiche: in tutta la simulazione, le scelte sui costi vengono effettuate in una modalità di fair play in cui ogni attore si comporta al meglio di quello che può. Infatti, lo scopo dei trasportatori è rientrare nei costi dello spostamento merci, quello dei mittenti di non spendere più di una certa cifra e quello dei nodi di prendere la loro commissione come intermediari. Questo approccio, come si vedrà nelle conclusioni, è un po' limitante rispetto al mercato reale dei trasporti.

2.3.2 Analisi di un time-step

La simulazione è organizzata secondo un ambiente a tempo discreto. Le operazioni vengono divise in time-step che corrispondono ognuno ad un certo numero di ore che possono essere settate nel codice. In questo modo, in ogni istante preso in considerazione vengono generati dei carichi nei nodi e vengono eseguite le aste per i carichi e i trasportatori già presenti. Le attività svolte nell'environment possono essere riassunte in questo modo:

Generazione dei carichi: in ogni time-step vengono generati carichi nei vari nodi seguendo una distribuzione di Poisson creata a partire dai database ETIS+ per quanto riguarda il numero di carichi in un particolare nodo. Questi carichi, come etto precedentemente, hanno una partenza ed una destinazione e vengono caricati in una lista ordinata di attesa. La destinazione viene sempre generata seguendo una tabella di probabilità generata sempre a partire da database ETIS+.

Aste: ogni carico viene bandito all'asta e, possibilmente, viene vinto da un trasportatore che viene pagato dal mittente.

Trasporto: a questo punto ogni carico viene trasportato. La distanza tra un nodo e un altro viene misurato in time-step, quindi nel time-step successivo alcuni

trasportatori saranno ancora in transito ed altri saranno arrivati in un altro nodo per ricominciare la catena di aste.

2.3.3 Fase di learning

La fase antecedente alla simulazione, denominata fase di apprendimento (learning), è cruciale poiché sia i trasportatori che i mittenti devono acquisire la capacità di interagire attraverso il meccanismo delle aste. La simulazione vera e propria avviene solo quando il numero di trasportatori distribuiti sulla rete raggiunge una quantità sufficiente. Questa fase di apprendimento è articolata in tre componenti chiave:

Inizializzazione: Si genera un mercato sovradimensionato, con un eccesso di trasportatori rispetto alla domanda. Durante 2500 time-steps, i nodi devono calcolare i pesi per ciascun collegamento, mentre i trasportatori apprendono il costo medio di permanenza in un nodo.

Valutazione dei trasportatori: Dopo il primo periodo di apprendimento, si cerca di eliminare i trasportatori in eccesso (coloro che generano un profitto nullo) e di aggiungerne nei nodi in cui l'asta viene vinta troppo spesso da un unico partecipante.

Ricalcolo dei pesi: Una volta raggiunto il numero adeguato di partecipanti, i nodi devono ricalcolare i pesi. Alla fine di questa fase, si dispone del numero corretto di trasportatori per la topologia di rete e si ottengono i pesi generati realisticamente dai nodi per ogni coppia origine-destinazione.

A partire da questi dati, si avvia la vera simulazione, avendo così creato il numero appropriato di trasportatori e i pesi necessari per alimentare il meccanismo delle aste.

2.3.4 SingleLane VS MultiLanes

Per poter validare i dati raccolti dalla simulazione, l'algoritmo proposto viene simulato nella stessa rete di nodi, con gli stessi dati in input in due modalità: quella singlelane e quella multilanes.

SingleLane: è l'approccio classico nel mondo della logistica ovvero il fatto di aggiudicare un trasporto dal punto di partenza o generazione al punto di destinazione. In questo modo, deve essere generato un cammino che attraversa più nodi per raggiungere la destinazione e quindi un unico trasportatore che vince l'asta e prende in carico il trasporto.

MultiLanes: l'algoritmo presentato crea un'asta per ogni passaggio della merce da un nodo ad un altro prevedendo la possibilità di affidare il carico a più trasportatori in base a quello più conveniente della tratta.

Dal punto di vista informatico, la differenza sta nel creare un cammino minimo completo a priori dal punto A al punto B, mentre nel secondo caso il percorso viene frammentato in più micropercorsi ognuno visto come il minimo locale. Nel caso della

simulazione si può vedere che questa strategia può abbassare i costi complessivi ed essere più efficiente a livello globale.

2.4 Gap di realizzazione

In questa sezione verranno presentati i maggiori gap di realizzazione tra il modello simulato e la realtà fisica della logistica. Alcuni sono presentati già dai realizzatori dell'algoritmo, altri sono considerazioni personali che derivano dallo studio dell'algoritmo in un contesto reale di un PI-hub.

2.4.1 Scopo simulazione

Bisogna partire dal presupposto che lo scopo della simulazione era testare l'effettiva fattibilità del routing tramite aste e di riuscire a verificare che sarebbe stato migliore di altri approcci. Partendo da questa idea, nonostante le limitazioni, il lavoro svolto dai ricercatori è stato esemplare, riuscendo ad imbastire un ambiente di test con dati realistici seppur artificiali che permettessero di analizzare i risultati puntualmente.

2.4.2 Fair Play degli attori

Nel contesto di un'applicazione pratica dell'algoritmo delle aste in un vero PI-hub emergono diverse sfide. Nel mondo reale, gli attori coinvolti seguono le regole del mercato, ma la loro condotta potrebbe non sempre rispettare criteri di correttezza. Data la natura intrinseca della ricerca di profitto, si potrebbero mettere in atto strategie finalizzate alla massimizzazione del guadagno. Ad esempio, i trasportatori potrebbero deliberatamente rinunciare a un carico per attendere un'asta più vantaggiosa. In alternativa, i mittenti potrebbero impostare un reserve price eccessivamente basso, non assegnare l'asta a nessun partecipante, e gradualmente aumentare il prezzo massimo fino a consegnarla al trasportatore che offre il costo effettivamente più basso, eludendo così la competizione equa. Questo comporterebbe problematiche legate ai tempi di svolgimento delle aste, creando congestionamenti sia di trasportatori in attesa prolungata negli hub, sia di merci in attesa, andando in controtendenza rispetto all'obiettivo dell'algoritmo di minimizzare i tempi operativi nei nodi. Pertanto, in un'implementazione pratica, è necessario adattare l'algoritmo delle aste per consentire aste rapide, ma che siano sensibili alle dinamiche del mercato e alle strategie adottate dagli attori coinvolti.

2.4.3 Il tempo nella simulazione

Come già spiegato, l'ambiente di simulazione è costruito con un modello a tempo discreto che in questo caso svolge il compito di poter simulare il passaggio di un

numero predefinito di ore. In queste ore, vengono automaticamente generati i carichi nei vari nodi e i trasportatori hanno tempo di spostarsi tra un nodo e un altro in modo che in ogni time-step siano sempre disponibili nei nodi un certo numero di nuovi carichi da bandire all'asta e un certo numero di trasportatori in attesa di partecipare in modo da creare concorrenza. Nel caso reale i trasportatori arrivano scaglionati e inoltre potrebbero non poter partecipare subito ad un'asta per questioni di tempistiche relative alle ore di riposo del conducente oppure di scaricamento merci. In aggiunta, il processo di assegnazione di un carico quando soltanto un trasportatore partecipa all'asta si svolge quasi istantaneamente. Tuttavia, il mittente potrebbe preferire attendere un momento in cui vi sia una maggiore concorrenza, al fine di poter ottimizzare i costi e spendere meno. In ultimo, dato che il trasportatore o meglio il personale della ditta di trasporti dovrà proporre un prezzo per ogni strada percorribile questo dovrebbe prevedere un certo ritardo perché non per forza dev'essere gestita da una procedura automatica. Quindi nella realizzazione sarebbe auspicabile inserire un meccanismo di timing nelle aste, che non sarebbero più immediate, ma che tengano conto dell'intervento umano lato trasportatore e lato mittente.

2.4.4 Gap nei vincoli

Nella sezione precedente, sono stati esaminati i principali vincoli introdotti dall'autore della simulazione. Va notato che non tutti questi vincoli generano disparità nell'implementazione, ma sono stati introdotti per semplificare il problema. Per esempio, la considerazione di 11 nodi fissi non costituisce un problema significativo, in quanto l'algoritmo può essere adattato replicandolo con l'aggiunta o la rimozione di nodi, nonché la creazione o l'eliminazione di archi di collegamento. Allo stesso modo, la limitazione a un singolo mittente che genera tutti i carichi nei nodi è facilmente correggibile durante l'implementazione, poiché l'algoritmo si occupa dei carichi, non dei mittenti. In pratica, questa scelta ha ristretto il caso di studio per motivi di prestazione della simulazione. Tuttavia, altri vincoli creano disparità nell'implementazione poiché influiscono sull'algoritmo delle aste. Ad esempio, semplificare il singolo trasportatore a un Full Truck Load, stabilendo un'assegnazione uno a uno tra carico generato e trasportatore, ha ripercussioni significative nel mondo reale. In generale, i trasportatori agiscono come container nella PI, e si presume che possano caricare merci provenienti da più mittenti per ottimizzare lo spazio disponibile e massimizzare il guadagno derivante dagli spostamenti. Tuttavia, la restrizione a un rapporto uno a uno con la merce impedisce all'algoritmo di considerare che lo stesso trasportatore possa partecipare a più aste in modo ottimizzato, consentendo di lasciare il PI-hub con il mezzo completamente carico. Infine, come discusso in precedenza, l'algoritmo richiederebbe modifiche per quanto riguarda il comportamento degli attori. Affinché una versione reale del PI abbia successo, deve attirare sia chi trasporta che chi spedisce. Le aziende di trasporti, se non vedono un

profitto nell'aderire alla rete di PI, potrebbero continuare a operare autonomamente per massimizzare il profitto, anche a discapito dell'ottimizzazione o della riduzione delle emissioni. Allo stesso modo, i mittenti dovranno essere persuasi a utilizzare un PI-hub anziché un trasportatore diretto, e se non è vantaggioso per loro, potrebbero cercare alternative e non aderire alla rete. Infine, è importante sottolineare che un PI con pochi trasportatori e mittenti può funzionare solo a livello locale, mentre il vero potenziale di tale sistema si manifesta su scala globale.

2.4.5 Gap riconosciuti dagli ideatori

Le limitazioni identificate nell'algoritmo sono state esplicitamente riconosciute dagli sviluppatori stessi. Questo è dovuto al fatto che l'obiettivo iniziale non era la creazione di un algoritmo immediatamente implementabile, bensì la valutazione della sua efficacia rispetto al routing single-lane attualmente utilizzato nel campo della logistica.

Problemi di collegamento: Nella soluzione attualmente proposta, non si tiene conto di problemi comuni come il traffico, i guasti o l'impossibilità di percorrere determinati tratti stradali. Per affrontare queste situazioni, sarebbe necessario implementare una procedura di re-routing, indirizzando i trasportatori verso percorsi alternativi accessibili. Questa operazione inevitabilmente comporterebbe un aumento dei tempi di percorrenza tra un nodo e l'altro, con conseguente aumento dei costi di trasporto associati.

Tipologie di merci: il trattamento delle merci all'interno della simulazione avviene in maniera omogenea, senza considerare la diversità di requisiti che possono presentarsi. Non viene contemplata la possibilità di gestire materiali deperibili, che richiedono specifici metodi e tempistiche di trasporto, o merci pericolose, che possono essere trasportate solo tramite mezzi dedicati. La risoluzione di questa problematica può essere implementata relativamente facilmente, escludendo dalla selezione dei trasportatori per l'asta quelli non idonei al trasporto di determinati materiali. Tuttavia, persiste il rischio di carenza di mezzi particolari, che dovrebbero essere distribuiti in modo ottimale sulla rete. Tale problematica deve essere considerata attentamente nel caso si desideri integrare tali tipologie di merci nella rete del PI.

Ping-Pong nel tragitto: un problema emerso durante le esecuzioni della simulazione riguarda il movimento di merci che vengono, in modo ciclico, rimbalzate tra due nodi a causa della possibile presenza di un arco chiuso nel grafo tra il nodo di partenza e quello di destinazione. Questo accade poiché attualmente manca un controllo sul percorso completo durante l'asta e non è implementata una memoria degli archi già attraversati. Questa problematica può essere risolta integrando al sistema delle aste un meccanismo di calcolo dei percorsi sul grafo, garantendo che non vengano mai attraversati in senso opposto gli stessi archi, tranne in casi eccezionali

legati a problemi di traffico o strade inaccessibili. Tuttavia, è importante sottolineare che questa problematica è già riconosciuta nel contesto del DI che potrebbe costituire da fonte di ispirazione per lo sviluppo di un algoritmo complessivamente più efficiente.

2.4.6 Simulazione VS realtà

Nell'ambito della simulazione, ciascun attore è modellato e simulato sulla stessa macchina per motivi di semplificazione e prototipazione rapida. I nodi, i mittenti, gli archi di collegamento e i trasportatori sono tutti definiti nell'ambiente di simulazione. Sebbene questa integrazione sia essenziale per il contesto simulato, è importante sottolineare che nella realtà questi attori sono distinti sia dal punto di vista logico che da quello fisico. Tale differenza rappresenta il principale divario che sarà oggetto di analisi nel modello proposto all'interno di questa tesi e più precisamente nel capitolo 4.

2.5 Conclusioni

In sintesi, in questo capitolo sono state esaminate a fondo le caratteristiche degli algoritmi di simulazione proposti nella pubblicazione, comprendendone il funzionamento, l'implementazione del codice e le relative limitazioni. Come inizialmente detto, i risultati ottenuti costituiscono una preziosa base per la considerazione del routing all'interno del PI, includendo una logica di pagamento attraverso aste, un elemento fino ad ora assente. Dal punto di vista tecnico, alcune lacune possono essere risolte con relativa facilità, mentre altre richiedono soluzioni più laboriose. In ogni caso, questo approccio rappresenta un eccellente punto di partenza per l'implementazione del routing e dei sistemi di pagamento all'interno del PI.

Capitolo 3

Metodologie

In questo capitolo vengono esposte le metodologie utilizzate per la stesura di questo testo a partire dall'approccio iniziale fino ad arrivare all'analisi approfondita del problema proposto.

3.1 Pubblicazione di partenza

La prima analisi effettuata è partita dalla lettura della pubblicazione presentata nel capitolo 2 e dalla successiva analisi del codice in Python della simulazione del sistema ad aste che è disponibile open source su Github [8]. Il codice sorgente si articola in tre componenti fondamentali, ciascuna dedicata a un preciso compito. In primo luogo, troviamo l'algoritmo delle aste, che gestisce il processo di assegnazione dei carichi ai trasportatori attraverso il meccanismo di aste. In secondo luogo, l'algoritmo di training svolge un ruolo cruciale nel preparare gli attori della simulazione, inizializzando i loro parametri a partire da dati reali. Infine, l'algoritmo di simulazione a tempo discreto, chiamato 'game', coordina lo sviluppo e l'evoluzione dell'ambiente di simulazione nel corso del tempo. In aggiunta, è inclusa una cartella contenente dei file R Markdown (rmd) che agevolano la visualizzazione dei risultati delle simulazioni pregresse effettuate dagli autori del codice. Ciò risponde alla necessità di una fase preliminare di training e apprendimento del modello, eseguita prima dell'avvio della simulazione che richiede una considerevole potenza di calcolo, non facilmente accessibile da tutti i dispositivi. Per dare un'idea delle risorse coinvolte, alcuni script richiedono più di 64 GB di RAM per la loro esecuzione.

3.2 Manipolazione codice

La prima fase nell'esame del codice, oltre all'analisi degli algoritmi, ha consistito nel tentativo di preparare l'ambiente per sfruttare i dati generati dalla simulazione eseguita dagli autori. L'obiettivo è stato comprendere come adattare il progetto

esistente per poterlo utilizzare nella conduzione di ulteriori simulazioni basate su dati reali, anziché su dati generati a partire da statistiche. Questa situazione ha rappresentato una sfida significativa, poiché attualmente manca una rete fisica dedicata per la validazione di qualsiasi risultato di ricerca e studio nell'ambito del PI.

3.2.1 Notebook Jupyter

I dati da analizzare vengono generati attraverso l'uso di Jupyter, fornendo una presentazione visuale dei risultati delle simulazioni. L'esecuzione di tali analisi è agevolata mediante l'installazione dell'ambiente, con la produzione di file PDF contenenti i grafici presenti nella pubblicazione. Questa modalità è stata resa accessibile per consentire la visualizzazione dei risultati della simulazione anche a coloro che potrebbero non disporre delle risorse necessarie per eseguire autonomamente le simulazioni.

3.2.2 Esecuzione simulazione

Dal punto di vista pratico, riducendo il numero di nodi e limitando la simulazione ai casi in cui il nodo non richiede commissioni, è possibile eseguire la simulazione su un comune PC in circa 5 minuti. Tuttavia, è importante sottolineare che, sebbene questa configurazione sia idonea per scopi statistici e valutativi dell'algoritmo, produce dati limitati riguardo alle sfide operative. Ad esempio, la frequenza con cui un singolo trasportatore partecipa alle aste, la circolazione senza meta di alcune merci o il tempo medio impiegato su una specifica tratta sono aspetti che, se implementati nella simulazione, potrebbero fornire dati più dettagliati e informativi per guidare eventuali modifiche e ottimizzazioni.

Capitolo 4

Modello proposto

In questo capitolo, verrà presentato il modello che è stato creato a partire dalla simulazione presente su Github relativa al meccanismo ad aste per provare a dare una prima versione dell'algoritmo, seppur semplificato, per cercare di risolvere uno dei principali gap di realizzazione e un modello informatico più aderente alla realtà.

4.1 Simulazione vs realtà

Come già anticipato, questo lavoro si propone di affrontare direttamente uno dei principali ostacoli nell'adattare la simulazione e il codice illustrato nei capitoli precedenti per renderli applicabili in contesti reali. Il principale problema di un adattamento diretto sta nella vocazione puramente simulativa del codice, il che implica che il programma disponibile non può essere semplicemente applicato alla realtà senza opportune modifiche. Sotto il profilo del codice proposto, tutte le entità sono modellate all'interno di un ambiente simulato, un'environment che, nella realtà, coincide proprio con l'ambiente reale. I nodi sono i PI-hubs, i collegamenti sono le strade, le linee ferroviarie, aeree o navali e i trasportatori e i mittenti sono reali aziende di logistica e di invio merci. Inoltre, sia i mittenti che i trasportatori nel momento attuale possono essere modellati in modo molto eterogeneo. Ad esempio, l'invio di merci può essere gestito da un'azienda fisica, piuttosto che da un e-commerce o anche da semplici privati. Le aziende di trasporto possono essere di qualsiasi tipologia, aziende di logistica internazionale, nazionale, ma anche locale con mezzi e restrizione di trasporto diversi. La distinzione chiave tra simulazione e realtà risiede principalmente nella scala e nella distribuzione degli attori che rendono operativo l'algoritmo di routing basato su aste. Inoltre, ogni nodo, trasportatore e mittente è fisicamente isolato dagli altri, richiedendo pertanto un adattamento del modello al fine di tener conto di tali vincoli.

4.2 Il modello della simulazione

In questa sezione, verranno analizzate le caratteristiche del modello del codice della simulazione presentato nel capitolo 2 che più di tutte non possono funzionare nel cercare di adattare il codice originale al caso reale.

4.2.1 Sistema centralizzato

Il modello di simulazione opera in modo centralizzato con la classe base Environment, che funge da contenitore e orologio per tutte le altre classi. All'interno di questa classe, si trova la lista di nodi, trasportatori, carichi generati e anche del mittente, che in questo caso è unico. Questa configurazione rappresenta una delle limitazioni della simulazione poiché, come evidenziato nella sezione precedente, la realtà si presenta in maniera significativamente differente. Sia dal punto di vista logico che fisico, tutte queste entità sono separate e probabilmente distanti tra loro. Pertanto, è essenziale considerare la gestione della comunicazione tra di esse attraverso la creazione di un sistema distribuito, tenendo conto del fatto che ogni attore interagisce con il sistema tramite un'interfaccia verso un sistema distribuito.

4.2.2 Organizzazione dei nodi

I nodi sono modellati tramite una lista e utilizzano i pesi precalcolati durante la fase di learning della simulazione leggendoli da un file. Questo tipo di struttura potrebbe essere più agevolmente modellata tramite un grafo pesato che permetta di mantenere memoria della topologia di rete e di utilizzare algoritmi noti in fase di test per agevolare alcune operazioni oppure ad esempio per poter gestire il re-routing delle merci in caso di interruzione di un arco.

4.2.3 Memoria

L'esecuzione attuale della simulazione non mantiene alcuna informazione in memoria, poiché si basa su pesi precalcolati e dati statici. Per renderla più realistica, sarebbe necessario considerare l'implementazione di uno o più database. Oltre a gestire la memoria della topologia di rete, questi database sarebbero essenziali per trasformare il codice in un sistema distribuito, consentendo la condivisione di dati tra tutti gli attori coinvolti. Inoltre, l'adozione di un database a grafo per la topologia di rete potrebbe fornire notevoli vantaggi nella gestione di strutture di questo tipo.

4.2.4 Hardware

Un ulteriore ostacolo riscontrato nella possibilità di adottare il codice fornito e sperimentare ottimizzazioni, o persino modificarlo per utilizzare topologie diverse,

è rappresentato dai requisiti hardware necessari per l'esecuzione. Entrambe le fasi, sia il learning che le fasi successive, richiedono un hardware con almeno 64 GB di RAM, quindi non un semplice PC. La sola modalità di esecuzione testata sembra funzionare riducendo drasticamente il numero di nodi a soli tre. Tuttavia, questa approccio, sebbene riduca notevolmente il numero di operazioni, potrebbe rendere i risultati poco attendibili o comunque meno significativi.

4.2.5 Conclusioni

Per tutti i motivi sopra elencati, è stato deciso di cambiare drasticamente approccio rispetto al codice della simulazione per provare a risolvere alcuni dei problemi che sono emersi in questa sezione.

4.3 Tecnologie modello proposto

In questa sezione, esamineremo le tecnologie e la teoria adottate per presentare il nuovo modello, approfondendo le ragioni delle soluzioni tecnologiche selezionate.

4.3.1 Mock object

I mock object sono oggetti simulati che vengono utilizzati nei test software per simulare il comportamento di oggetti reali in un ambiente di sviluppo. Questi oggetti simulati vengono creati per emulare il comportamento di componenti del software o del sistema che non sono disponibili o non sono pratici da utilizzare in un contesto di test. L'obiettivo principale dei mock object è isolare la componente specifica che si sta testando, consentendo ai programmatori di concentrarsi solo sulla logica della funzione o del modulo che stanno sviluppando. I mock object possono simulare risposte positive o negative, errori o qualsiasi comportamento desiderato, consentendo ai programmatori di verificare come il codice interagisce con le dipendenze senza dover utilizzare implementazioni reali di tali dipendenze. I mock object sono particolarmente utili nei test unitari e nei test di integrazione, dove è importante isolarli da componenti esterne al fine di garantire test più affidabili e ripetibili.

4.3.2 I mock object nel modello

Nel modello è stata adottato l'approccio di utilizzare due mock object per rappresentare due attori principali della simulazione: il nodo responsabile di bandire le aste e il trasportatore partecipante. Queste due entità agiscono come mock object in quanto simulano, nel caso del nodo, la creazione di un'asta, e nel caso del trasportatore, la risposta a un'asta. Essendo mock object il codice dietro a questi due attori non è implementato, ma solo simulato. Nel dettaglio, la generazione di un'asta è simulata nel caso del nodo, mentre la partecipazione è simulata da una

risposta con un valore casuale da parte del trasportatore. È importante sottolineare che questo studio non si focalizza sui dettagli implementativi delle aste o del routing, aspetti già trattati in modo approfondito nella pubblicazione di riferimento. Piuttosto, l'attenzione è rivolta al meccanismo di interazione implementabile tra i vari attori in un contesto reale e distribuito.

4.3.3 Distribuzione e concorrenza del modello

Come anticipato, il modello studiato deve per forza di cose essere pensato in modo distribuito perché fisicamente ogni nodo è a sé e bandisce le sue aste, ogni trasportatore partecipa in concorrenza e ogni mittente genera i carichi contemporaneamente e su più nodi. Per gestire un sistema con queste funzionalità, bisogna cercare una soluzione distribuita e concorrente per poter gestire tutte le interazioni tra le parti. In questo modello si è tenuto conto solo dell'interazione tra nodo e trasportatore, ma il modello di comunicazione adottato può essere esteso anche a tutti gli altri componenti, tra nodo e nodo e tra nodo e mittente.

4.3.4 Nodo di riferimento

Per la selezione del protocollo di comunicazione in questo contesto, è stata presa in considerazione l'infrastruttura PI-hub situata presso il Politecnico di Torino, nel Dipartimento di Ingegneria Gestionale. Il laboratorio ospita un hub fisico progettato per automatizzare la gestione di cassette, consentendone lo spostamento e il riconoscimento della posizione. A causa di questa conoscenza e della familiarità con il software impiegato nel laboratorio, la scelta è ricaduta sul protocollo MQTT, ritenuto facilmente implementabile in un ambiente di questo tipo.

4.3.5 MQTT

MQTT[10], Message Queuing Telemetry Transport, è un protocollo di messaggistica leggero, aperto e basato sul modello di pubblicazione/sottoscrizione (pub/sub) costruito sopra la pila TCP/IP.

Figura 4.1: Logo MQTT

È stato progettato per essere efficiente, affidabile e facile da implementare, rendendolo particolarmente adatto per scenari in cui è necessario lo scambio di messaggi tra dispositivi con risorse limitate, in particolare per i dispositivi IoT. Il

pattern pub/sub richiede un broker di messaggistica intermedio che si occupa della distribuzione dei messaggi ai destinatari.

Vengono qui elencate le caratteristiche chiave utili a comprendere l'algoritmo di comunicazione tra nodo e trasportatore.

Modello di pubblicazione/sottoscrizione: MQTT si basa su un modello di comunicazione in cui i dispositivi possono essere editori (publishers) di messaggi o sottoscrittori (subscribers) che ricevono i messaggi. Gli editori inviano messaggi a specifici argomenti (topics) e i sottoscrittori ricevono messaggi relativi agli argomenti a cui sono interessati.

Leggerezza e efficienza: MQTT è progettato per essere leggero e richiedere una quantità minima di larghezza di banda e poche risorse di sistema. Questo lo rende adatto per dispositivi con limitate risorse computazionali e connessioni di rete a bassa larghezza di banda oppure per comunicazioni a bassa latenza.

Qualità del servizio (QoS): MQTT offre tre livelli di QoS per garantire la consegna affidabile dei messaggi:

QoS 0: Consegna al più una volta (nessuna conferma).

QoS 1: Consegna almeno una volta (con conferma).

QoS 2: Consegna esattamente una volta attraverso un handshake a tre passaggi.

Salvataggio dei messaggi: MQTT consente di memorizzare i messaggi dei vari topics in modo che i nuovi sottoscrittori possano ricevere l'ultimo messaggio pubblicato sull'argomento quando si connettono.

Persistenza della sessione: MQTT consente alle sessioni di rimanere persistenti anche dopo la disconnessione di un client, garantendo che possa ricevere i messaggi che ha perso durante la disconnessione quando si riconnette.

Scalabilità: MQTT è altamente scalabile e può essere utilizzato in reti con un grande numero di dispositivi. È progettato per funzionare in scenari in cui la topologia di rete è dinamica (esattamente come nel caso dei trasportatori presenti in un nodo). MQTT è ampiamente utilizzato nell'ambito dell'IoT, automazione domestica, monitoraggio remoto e in molti altri contesti in cui è necessario uno scambio efficiente e affidabile di messaggi tra dispositivi distribuiti. La sua adozione è favorita dalla sua semplicità e versatilità.

Il broker MQTT è un server che funge da intermediario nella comunicazione tra i dispositivi che utilizzano il protocollo e gestisce la ricezione e l'inoltro dei messaggi tra i vari client MQTT presenti nella rete. Le sue funzioni principali sono:

Accettazione dei Messaggi: il broker riceve i messaggi inviati dai client MQTT e ne gestisce l'arrivo.

Routing dei messaggi: Il broker instrada i messaggi ai client destinatari sulla base dei topics a cui sono iscritti.

Memorizzazione dei messaggi: in alcuni casi, il broker può memorizzare i messaggi in modo temporaneo, garantendo che i client ricevano i messaggi anche se non sono online al momento dell'invio.

Sottoscrizioni: il broker mantiene un elenco delle sottoscrizioni di tutti i client, consentendo una corretta distribuzione dei messaggi in base ai topic sottoscritti.

Autenticazione: il broker può richiedere un meccanismo di autenticazione per permettere la comunicazione solo tra client autorizzati a collegarsi tra loro.

Le porte standard del protocollo MQTT sono TCP/8883 (utilizzando TLS) e UDP/1883 (per comunicazioni in chiaro). Ovviamente se il passaggio di dati avviene tramite una rete pubblica è auspicabile l'utilizzo della versione sicura di MQTT sulla porta 8883 piuttosto che la comunicazione in chiaro UDP che, seppure più snella, può trovare utilizzo solo in reti private oppure in fase di test o prototipazione.

4.3.6 Perché MQTT

La decisione di adottare il protocollo MQTT per questo progetto si fonda principalmente su due motivi:

Ambiente di riferimento: basandoci sulle tecnologie impiegate nel laboratorio del Politecnico menzionato in precedenza, il PI può essere parzialmente modellato come un sistema IoT, in cui numerosi dispositivi collegati in rete eseguono porzioni di codice e comunicano tra loro e con uno o più server centrali. L'MQTT è stato considerato una scelta adatta per la comunicazione, poiché è stato progettato per la distribuzione di messaggi veloci con un basso utilizzo di banda nei sistemi IoT.

Velocità di prototipazione: dato che il protocollo MQTT è molto semplice da implementare in Python, richiedendo solo la configurazione di un broker intermedio e la creazione dei topics a piacimento, la scrittura del codice di comunicazione è significativamente più rapida rispetto ad altre soluzioni comunemente utilizzate nei programmi distribuiti e multithread.

In conclusione, la scelta è stata guidata dalla volontà di semplificare il processo di sviluppo, insieme alla possibilità futura di sostituire al mock del nodo il PI-hub reale, data la presenza di sistemi già implementati tramite MQTT nel laboratorio, senza dover modificare il codice relativo ai trasportatori che partecipano alle aste.

4.4 Il codice

4.4.1 Ambiente di esecuzione

Il codice è stato sviluppato in Python ed è eseguito in un virtual environment su Windows 11, ma può essere utilizzato nello stesso modo negli altri sistemi operativi, utilizzando il comando giusto per creare l'ambiente virtuale e installando i pacchetti necessari per l'esecuzione. Di seguito elencherò la procedura per Windows, ma può essere facilmente replicata in altri ambienti.

In un terminale, spostarsi nella cartella del progetto ed eseguire i seguenti comandi:

```
pip install virtualenv
python<versione> -m venv <nome-ambiente>
env/Scripts/activate.bat //In CMD
env/Scripts/Activate.ps1 //In Powershell
```

Listato 4.1: Comandi bash per ambiente di esecuzione in Windows

Una volta attivato il virtual environment, si possono andare ad installare tutti i pacchetti necessari per l'esecuzione del codice. La scelta di creare un virtual environment è stata presa per isolare i package presenti nel progetto da quelli dell'ambiente completo installato sulla macchina. Questo serve per evitare conflitti tra i package già installati per ogni progetto ed è auspicabile farlo ogni volta che si gestisce un progetto che prevede una complessità di file maggiore del singolo script Python.

4.4.2 Broker MQTT: HiveMQ Cloud

Nonostante sia possibile implementare soluzioni utilizzando server MQTT privati, la scelta è stata orientata verso HiveMQ Cloud[6] per ragioni di comodità ed affidabilità. HiveMQ Cloud è una piattaforma cloud che offre la possibilità di creare broker MQTT senza la necessità di configurare e mantenere un'infrastruttura privata più complessa.

Figura 4.2: Logo HiveMQ Cloud

Questo broker consente di connettere dei client MQTT con il semplice requisito di una connessione a Internet, agendo come intermediario e dispatcher dei messaggi MQTT. Inoltre, offre la possibilità di stabilire connessioni sicure mediante il protocollo TLS e autenticazione degli account dei client. La configurazione può essere effettuata agevolmente attraverso un'interfaccia web. Tale approccio semplifica notevolmente l'implementazione e la manutenzione del sistema, consentendo una maggiore focalizzazione sulle funzionalità specifiche dell'applicazione senza dover gestire l'infrastruttura sottostante. In aggiunta, poiché il codice degli attori coinvolti non è vincolato al codice della comunicazione, che funge da middleware, è possibile effettuare una sostituzione agevole del broker MQTT senza dover modificare la struttura del codice. Questa flessibilità consente la transizione fluida da

HiveMQ Cloud ad un altro servizio web o ancora ad un'installazione su server privati, mantenendo intatto il funzionamento degli attori coinvolti. La modularità del sistema permette, dunque, di adattarsi facilmente a diversi contesti e requisiti di infrastruttura senza richiedere modifiche sostanziali al codice sorgente.

4.4.3 Middleware

Come detto precedentemente, il modello di comunicazione tramite MQTT rappresenta un middleware per separare fisicamente e logicamente nodo e trasportatori che partecipano all'asta tramite un canale di comunicazione comune tramite messaggi. In questo modo, il codice e le operazioni logiche eseguite dai vari attori sono disaccoppiati e possono essere gestiti autonomamente dai vari partecipanti. Questa interfaccia può essere replicata per ogni attore presente nel modello e potrebbe essere implementato anche con altri protocolli in un secondo momento. Questa è la chiave principale che rappresenta un primo tentativo di miglioramento per portare il codice della simulazione a funzionare in un ambiente reale.

4.4.4 Schema del modello

Il seguente schema riassume il modello di interazione implementato nel codice prodotto.

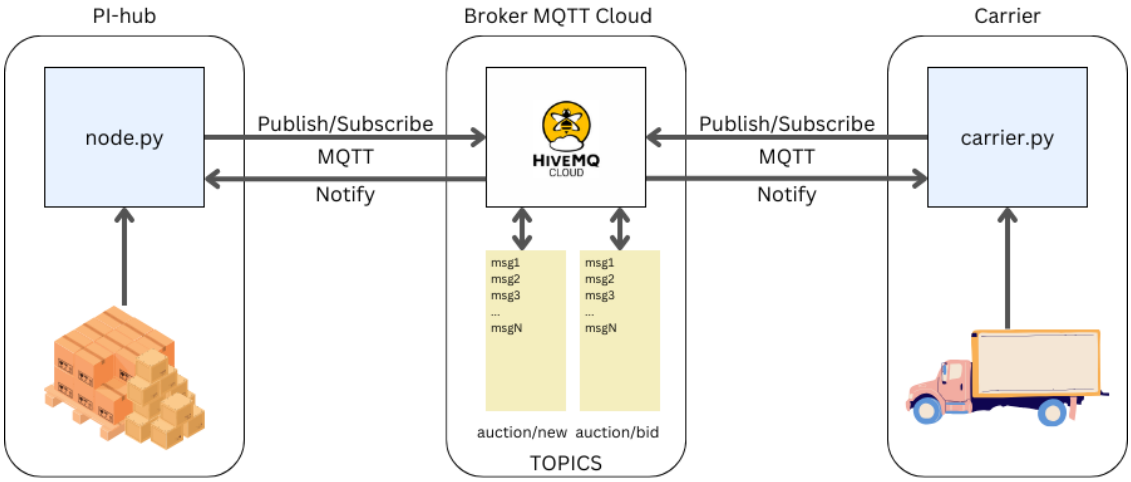

Figura 4.3: Architettura della soluzione proposta

Il nodo esegue il codice del file node.py che si iscrive al topic "auction/bid" e pubblica sul topic "auction/new" quando bandisce una nuova asta. Al contrario, il trasportatore tramite il codice carrier.py si iscrive al topic "auction/new" per ricevere una notifica quando viene pubblicata una nuova richiesta di asta e pubblica

la sua offerta nel topic "auction/bid/<id auction>". In questo modo, si crea un nuovo topic per ogni asta, in modo da poter raggruppare in futuro le offerte di tutti i carriers che partecipano alla stessa asta, senza andare a leggere il corpo del messaggio per capire l'identificativo dell'asta.

Al centro si trova il broker MQTT, in questo caso HiveMQ Cloud, che funge da intermediario tenendo traccia dei client che si sono iscritti ai vari topic, occupandosi di ricevere i messaggi organizzandoli in topic e notificando i nuovi messaggi ai client iscritti.

Nella figura 4.4, si può invece vedere il dettaglio delle operazioni temporizzate di un'asta tramite il client fornito dal paccheto Python paho.mqtt.client. Quello rappresentato è il caso notevole, presente anche nel codice, dell'interfacciamento tra un nodo ed un unico trasportatore che partecipa all'asta. Lo stesso modello, data la presenza del middleware fornito dal broker intermedio, può essere applicato nel caso di n trasportatori partecipanti all'asta.

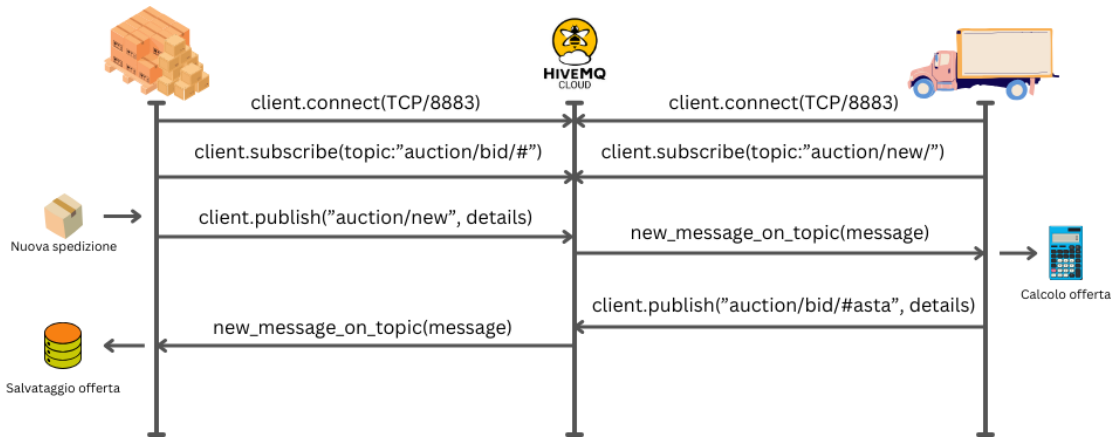

Figura 4.4: Schema temporale di un'asta

4.4.5 Codice del nodo

```
1  import time
2  import paho.mqtt.client as paho
3  from paho import mqtt
4
5  # setting callbacks
6  def on_connect(client, userdata, flags, rc, properties=None):
7      """
8          Prints the result of the connection with a
              reasoncode to stdout ( used as callback for
              connect )
```

```
9
10          :param client: the client itself
11          :param userdata: userdata is set when initiating the
                client, here it is userdata=None
12          :param flags: these are response flags sent by the
                broker
13          :param rc: stands for reasonCode, which is a code
                for the connection result
14          :param properties: can be used in MQTTv5, but is
                optional
15      """
16      print("CONNACK␣received␣with␣code␣%s." % rc)
17
18
19  # with this callback you can see if your publish was
        successful
20  def on_publish(client, userdata, mid, properties=None):
21      """
22          Prints mid to stdout to reassure a successful
                publish ( used as callback for publish )
23
24          :param client: the client itself
25          :param userdata: userdata is set when initiating the
                client, here it is userdata=None
26          :param mid: variable returned from the corresponding
                publish() call, to allow outgoing messages to be
                tracked
27          :param properties: can be used in MQTTv5, but is
                optional
28      """
29      print("mid:␣" + str(mid))
30
31
32  # print which topic was subscribed to
33  def on_subscribe(client, userdata, mid, granted_qos,
        properties=None):
34      """
35          Prints a reassurance for successfully subscribing
36
37          :param client: the client itself
38          :param userdata: userdata is set when initiating the
                client, here it is userdata=None
39          :param mid: variable returned from the corresponding
                publish() call, to allow outgoing messages to be
                tracked
```

```
40          :param granted_qos: this is the qos that you declare
               when subscribing, use the same one for publishing
41          :param properties: can be used in MQTTv5, but is
               optional
42      """
43      print("Subscribed: " + str(mid) + " " + str(granted_qos))
44
45  # print message, useful for checking if it was successful
46  def on_message(client, userdata, msg):
47      """
48          Prints a mqtt message to stdout ( used as callback
               for subscribe )
49
50          :param client: the client itself
51          :param userdata: userdata is set when initiating the
               client, here it is userdata=None
52          :param msg: the message with topic and payload
53      """
54
55
56      print(msg.topic + " " + str(msg.qos) + " " +
           str(msg.payload))
57
58
59
60  client = paho.Client(client_id="", userdata=None,
       protocol=paho.MQTTv5)
61  client.on_connect = on_connect
62
63  # enable TLS for secure connection
64  client.tls_set(tls_version=mqtt.client.ssl.PROTOCOL_TLS)
65  # set username and password
66  client.username_pw_set(<username_account>,
       <password_account>)
67  # connect to HiveMQ Cloud on port 8883 (default for MQTT)
68  client.connect(<url_servizio_cloud>, 8883)
69
70  # setting callbacks
71  client.on_subscribe = on_subscribe
72  client.on_message = on_message
73  client.on_publish = on_publish
74
75  # subscribe to the channel of bids
76  client.subscribe("auction/bid/#", qos=1)
77  start = 0
78  for x in range (200):
```

```
79
80      # calc and write the duration on file
81      end = time.time()
82      delta_time= end - start
83      print(delta_time)
84      f = open("res_time.txt", "a")
85      f.write(str(delta_time)+"\n")
86      f.close()
87      # a little unrealistic delay
88      time.sleep(1)
89      # new auction on the channel
90      msg = "new-auction-" + str(x)+"-"+str(time.time())
91      start = time.time()
92      client.publish("auction/new", payload=msg, qos=1)
93      # loop on the buffer
94      client.loop()
95
96
97  client.loop_forever()
```

Listato 4.2: Codice del nodo del file node.py

4.4.6 Codice del carrier

```
1  import time
2  import random
3  import paho.mqtt.client as paho
4  from paho import mqtt
5
6
7  # setting callbacks for different events to see if it works,
       print the message etc.
8  def on_connect(client, userdata, flags, rc, properties=None):
9      """
10         Prints the result of the connection with a
               reasoncode to stdout ( used as callback for
               connect )
11
12         :param client: the client itself
13         :param userdata: userdata is set when initiating the
               client, here it is userdata=None
14         :param flags: these are response flags sent by the
               broker
15         :param rc: stands for reasonCode, which is a code
               for the connection result
```

```
16          :param properties: can be used in MQTTv5, but is
                optional
17      """
18      print("CONNACK␣received␣with␣code␣%s." % rc)
19
20
21  # with this callback you can see if your publish was
        successful
22  def on_publish(client, userdata, mid, properties=None):
23      """
24          Prints mid to stdout to reassure a successful
                publish ( used as callback for publish )
25
26          :param client: the client itself
27          :param userdata: userdata is set when initiating the
                client, here it is userdata=None
28          :param mid: variable returned from the corresponding
                publish() call, to allow outgoing messages to be
                tracked
29          :param properties: can be used in MQTTv5, but is
                optional
30      """
31      print("mid:␣" + str(mid))
32
33
34  # print which topic was subscribed to
35  def on_subscribe(client, userdata, mid, granted_qos,
        properties=None):
36      """
37          Prints a reassurance for successfully subscribing
38
39          :param client: the client itself
40          :param userdata: userdata is set when initiating the
                client, here it is userdata=None
41          :param mid: variable returned from the corresponding
                publish() call, to allow outgoing messages to be
                tracked
42          :param granted_qos: this is the qos that you declare
                when subscribing, use the same one for publishing
43          :param properties: can be used in MQTTv5, but is
                optional
44      """
45      print("Subscribed:␣" + str(mid) + "␣" + str(granted_qos))
46
47
48  # callback for a new mqtt message
```

```
49 def on_message(client, userdata, msg):
50
51     print(msg.topic + "␣" + str(msg.qos) + "␣" +
          str(msg.payload))
52     numBid= str(msg.payload).split("-")
53     # little unrealistic delay
54     time.sleep(0.01)
55     bid = random.randint(1,10)
56     # response to the auction
57     client.publish("auction/bid/"+numBid[2],
          payload=str(bid)+"-"+numBid[3], qos=1)
58
59
60 client = paho.Client(client_id="", userdata=None,
      protocol=paho.MQTTv5)
61 client.on_connect = on_connect
62
63 # enable TLS for secure connection
64 client.tls_set(tls_version=mqtt.client.ssl.PROTOCOL_TLS)
65 # set username and password
66 client.username_pw_set(<username_account>,
      <password_account>)
67 # connect to HiveMQ Cloud on port 8883 (default for MQTT)
68 client.connect(<url_servizio_cloud>, 8883)
69
70 # callbacks for the operations mqtt
71 client.on_subscribe = on_subscribe
72 client.on_message = on_message
73 client.on_publish = on_publish
74
75 # subscribe to topic auction/new where the node will publish
      a new request for auction
76 client.subscribe("auction/new", qos=1)
77
78 client.loop_forever()
```

Listato 4.3: Codice del trasportatore del file carrier.py

4.5 Risultati e possibili miglioramenti

4.5.1 Risultato dello studio

Il risultato ottenuto è un modello snello e funzionante di scambio di messaggi tra nodo ed un trasportatore utilizzando dei mock per simulare il comportamento degli

attori. Inoltre, il codice produce in output la durata in millisecondi tra la pubblicazione di un'asta da parte del nodo e la risposta del trasportatore dopo un tempo minimo fissato di attesa dalla lettura della richiesta. Questi dati creano una distribuzione di tempistiche utili in futuro per poter analizzare la latenza del modello quando applicato ad un caso con componenti reali (ad esempio il PI-hub oppure il programma di partecipazione alle aste dei trasportatori). Queste latenze sono composte da un leggero ritardo che può essere modificato a piacimento e poi dal tempo di ricezione del messaggio notificato dal broker al trasportatore e il tempo di latenza di pubblicazione della risposta dal trasportatore al broker e la successiva notifica del broker al nodo. In questo caso le latenze sono bassissime perché l'esecuzione è stata svolta attivando i due processi su un unico PC collegato in wifi nella rete Polito dal magazzino verso il broker di HiveMQ con un overhead di pochi millisecondi. Nel caso di utilizzo con dispositivi IoT, magari collegati alla rete tramite SIM GSM oppure con altre tecnologie con poca larghezza di banda lo scenario può cambiare drasticamente, ma dato il leggero utilizzo di banda del protocollo MQTT dovrebbe comunque essere efficiente anche in questi casi.

4.5.2 Miglioramenti futuri

L'algoritmo attuale presenta diversi punti che possono essere migliorati. Inizialmente, l'approccio adottato non ha considerato la presenza di più di un trasportatore che partecipa all'asta. Una possibile evoluzione del modello di comunicazione mediante MQTT potrebbe essere implementare un'asta temporizzata, introducendo una scadenza che consenta ai trasportatori di presentare le loro offerte. Per fare ciò, sarebbe sufficiente apportare modifiche mirate al codice, chiudendo l'asta quando viene raggiunto un numero prestabilito di offerte o trascorso un intervallo di tempo specifico. Ulteriori ottimizzazioni potrebbero derivare dall'implementazione di un database centrale a grafo, che tenga conto dei pesi associati ai vari collegamenti percorribili da ciascun trasportatore. Questi pesi possono essere definiti in base alla distanza in chilometri e agli eventuali vantaggi derivanti dalla posizione del nodo o ancora dalla tipologia di mezzi a disposizione dell'azienda di trasporti. Per aumentare la flessibilità del sistema, sarebbe ottimo separare completamente il codice relativo a nodi e trasportatori da quello relativo alla comunicazione. Questo consentirebbe l'adozione di diverse tipologie di comunicazione, ad esempio attraverso un web service che gestisca il middleware, consentendo l'integrazione di comunicazioni via web, MQTT o socket. Questa diversificazione consentirebbe di coinvolgere partecipanti con tecnologie eterogenee, garantendo comunque una flessibile interoperabilità. In fase di test, potrebbe inoltre essere utile implementare l'algoritmo proposto nel paper per determinare il vincitore delle aste. Inoltre, si potrebbe esplorare la possibilità di adottare un sistema basato sul meccanismo Single-Lane anziché sul Multi-Lanes, al fine di valutare i benefici anche in scenari reali.

Eventualmente, il codice fornito può servire come punto di partenza per implementazioni che possano integrare componenti reali.

Capitolo 5

Conclusioni

Lo scopo di questa tesi può essere individuato in due obiettivi principali. Il primo, contribuire al body of knowledge in materia di protocolli di routing nel PI che attualmente è ancora in fase embrionale, studiandone gli algoritmi esistenti e analizzandone la fattibilità e i principali gap rispetto ad una realizzazione reale. Il secondo, di realizzare e analizzare uno strumento pratico che potesse fungere da mock object in una situazione reale prendendo come algoritmo di riferimento quello presentato nel paper di riferimento. Per poter individuare i requisiti per adattare tale algoritmo ad un caso reale, si è considerato come nodo di riferimento il sistema AS/RS (Automated Storage/Retrieval System) presente nel laboratorio RESLOG del Politecnico di Torino. Svolta la fase di analisi, si è capito che si poteva creare a partire dall'idea centralizzata presentata nella simulazione delle aste una soluzione ad-hoc, questa volta distribuita, che potesse essere utilizzata in un ambito reale come il PI-hub preso in considerazione. Il risultato è stato un modello di interazione in Python in cui il nodo esegue il proprio codice per bandire le aste e il trasportatore che vuole partecipare risponde tramite il proprio codice, in un'ottica di distribuzione e concorrenza. In questo modo si è ottenuta una disassociazione tra i due attori, non più oggetti all'interno di una simulazione, ma ognuno con il proprio codice che interagisce con l'altro tramite un middleware. Il protocollo di comunicazione scelto è stato MQTT per una questione di integrazione con l'ambiente considerato tenendo conto che è uno standard per le comunicazioni tra dispositivi IoT data la bassa occupazione di banda. Il risultato di questo lavoro è un programma funzionante che permette al nodo di bandire l'asta pubblicandola come topic nel broker MQTT e al trasportatore iscritto al medesimo topics di rispondere con un'offerta. Nodo e trasportatore sono stati modellati come mock di oggetti reali, per favorire la possibilità di sostituzione con il sistema AS/RS in futuri sviluppi. Inoltre, sono stati monitorati i tempi di overhead per l'utilizzo di questo modello basato sulla comunicazione di rete con un broker in cloud rispetto ai tempi praticamente nulli e poco realistici se eseguito in una simulazione con attori coesistenti nello stesso programma. Tenendo conto che questi tempi sono dell'ordine di pochi

millisecondi, il sistema potrà essere in futuro testato su dispositivi con accesso alla rete eterogeneo.

Come analizzato nel capitolo precedente, ci sono molti miglioramenti che possono essere attuati a partire da questo primo approccio:

Database: sarebbe utile modellare il sistema tramite database a grafo per ottimizzare gli algoritmi di routing e la possibilità di aggiungere dinamicamente nodi e collegamenti.

Algoritmi di routing: un altra possibile implementazione per il routing delle merci potrebbe sfruttare algoritmi già in utilizzo nel DI, utilizzando ad esempio tabelle di routing statiche o dinamiche per decidere il peso dei collegamenti e dirigere la scelta di una route completa o parziale in base alle metriche selezionate.

Concorrenza: il modello dovrebbe essere espanso a più trasportatori che partecipano alle aste implementando una piattaforma per la partecipazione concorrente e temporizzando le aste in modo da garantire un buon bacino di utenza.

In conclusione, il prodotto fornito potrà essere utilizzato come punto di partenza per future implementazioni che permetteranno finalmente di portare nella realtà l'architettura del PI.

Bibliografia

[1] «Benoit Montreuil Linkedin profile». In: (2023). URL: https://www.linkedin.com/in/benoit-montreuil-a399213/.

[2] Martin Briand, Rod Franklin e Mariam Lafkihi. «A dynamic routing protocol with payments for the Physical Internet: A simulation with learning agents». In: *Transportation Research Part E: Logistics and Transportation Review* 166 (2022), p. 102905. ISSN: 1366-5545. DOI: https://doi.org/10.1016/j.tre.2022.102905. URL: https://www.sciencedirect.com/science/article/pii/S1366554522002824.

[3] Jean-Yves Colin, Hervé Mathieu e Moustafa Nakechbandi. «A Proposal for an Open Logistics Interconnection Reference Model for a Physical Internet». In: *CoRR* abs/1904.05069 (2019). URL: http://arxiv.org/abs/1904.05069.

[4] Economist. «The Physical Internet: A Survey of Logistic». In: (2006). URL: https://www.economist.com/special-report/2006/06/17/the-physical-internet.

[5] Russell D. Meller Eric Ballot Benoit Montreuil. *The Physical Internet: the Network of Logistics Networks*. PREDIT, 2014.

[6] «HiveMQ Cloud». In: (2023). URL: https://www.hivemq.com/.

[7] «ISO/OSI Wikipedia». In: (2023). URL: https://it.wikipedia.org/wiki/Modello_OSI.

[8] M.Briand. «PI routing protocol simulation». In: (2021). URL: https://github.com/MartinBriand/PI-routing_protocol_simulation/releases/tag/v2.0.0.

[9] B. Montreuil, E. Ballot e F. Fontane. «An Open Logistics Interconnection model for the Physical Internet». In: *IFAC Proceedings Volumes* 45.6 (2012). 14th IFAC Symposium on Information Control Problems in Manufacturing, pp. 327–332. ISSN: 1474-6670. DOI: https://doi.org/10.3182/20120523-3-RO-2023.00385. URL: https://www.sciencedirect.com/science/article/pii/S1474667016331718.

[10] «MQTT official website». In: (2023). URL: https://mqtt.org/.

[11] «Physical Internet Manifesto». In: (2012). URL: https://www.slideshare.net/physical_internet/physical-internet-manifesto-eng-version-1111-20121119-15252441.