



TRACE

inTegration & haRmonizAtion
of logistiCs opERations

D4.3 Synchromodal operations and optimization of shared resources (A)

Horizon Innovation Actions | Project No. 101104278

Call HORIZON-CL5-2022-D6-02



Co-funded by
the European Union

Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or CINEA. Neither the European Union nor CINEA can be held responsible for them

Dissemination level	Public (PU)
Type of deliverable	R – Document, report
Work package	WP4 – Infrastructure & Ecosystem
Status - version, date	Draft v1.0, 30/11/2024
Deliverable leader	UTH
Contractual date of delivery	30/11/2024
Actual date of delivery	30/11/2024

List of authors

Author Name	Organization
Anna Vrani, Savvas Apostolidis, Athanasios Petsanis, Manolis Raptis, Alexandra Kiziridou, Athanasios Kapoutsis, Elias Kosmatopoulos	CERTH
Theofilos Triommatis, Markos Papageorgiou, Ioannis Papamichail	TUC
Kolomvatsos Kostas, Kylafas Christos, Fountas Panagiotis	UTH
Tyimplalexis Nikolaos, Papadopoulou Peggy	UNISYSTEMS
Fazal Raheman, Tejas Bhagat	BC5
Vasilis Papataxiarxis, Anestis Papakotoulas	NKUA

Version History

Version	Date	Author	Description of changes
V0.1	15.09.2024	Kolomvatsos Kostas, Kylafas Christos, Fountas Panagiotis, Tymplalexis Nikolaos	Table of Contents specified
V0.2	14.10.2024	Kolomvatsos Kostas, Kylafas Christos, Fountas Panagiotis, Tymplalexis Nikolaos	First round of inputs requested from involved partners
V0.3	21.10.2024	Kolomvatsos Kostas, Kylafas Christos, Fountas Panagiotis, Tymplalexis Nikolaos	First round of input collected
V0.4	25.10.2024	Kolomvatsos Kostas, Kylafas Christos, Fountas Panagiotis, Tymplalexis Nikolaos	Second round of inputs requested from involved partners
V0.5	9.11.2024	Kolomvatsos Kostas, Kylafas Christos, Fountas Panagiotis, Tymplalexis Nikolaos	Second round of input collected and integrated
V0.6	17.11.2024	Kolomvatsos Kostas, Kylafas Christos, Fountas Panagiotis, Tymplalexis Nikolaos	Version for internal review
V0.7	26.11.2024	Kolomvatsos Kostas, Kylafas Christos, Fountas Panagiotis, Tymplalexis Nikolaos	Comments Addressed
V0.8	30.11.2024	Kolomvatsos Kostas	Submission

Peer Review

	Reviewer Name	Organization	Date
V0.7	Savvas Apostolidis, Anna Vrani	CERTH	26.11.2024
V0.6	Sheila Rodriguez, Christian Vasquez	ROBOTNIK	28.11.2024

Quality Manager Review

	Reviewer Name	Organization	Date
V1.0	Ioannis Neokosmidis	INCITES	30.11.2024

Legal Disclaimer

The information in this document is provided “as is”, and no guarantee or warranty is given that it is fit for any specific purpose. The TRACE project Consortium members shall have no liability for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials subject to any liability which is mandatory due to applicable law.

1 Executive Summary

This deliverable outlines the progress and operational structures with regard to synchromodal logistics and resource optimization within the TRACE project. Targeting logistics experts and relevant stakeholders, this document sets forth a wide-based methodology for enhancing operational efficiency through innovative technologies, such as event management systems, smart contracts based on blockchain, and advanced services in routing and scheduling optimization. It acts as a base reference for the conception and implementation of instruments meant for efficient resource management in collaborative logistics activities. The following are some of the core sections of the deliverable. The Introduction gives an overview of the goals and structure of the deliverable. The section Synchromodal Operations and Shared Resources describes in detail the definition, scope, and interaction between the diverse components of the logistic system, such as REM, MOM and FMM. This section further provides an analysis of the benefits and impacts of the modules with regard to their contribution in resource management and mitigation measures.

The Intelligent Services for Logistics Optimization section works on planning, scheduling, and routing of logistics, first, middle and last mile operations, going into detail about optimal pathfinding algorithms for ground and aerial vehicles. This discussion will lay out the integration of these services into TRACE to ensure that it meets all the architectural and project requirements. The section on blockchain infrastructure for smart contracts elaborates on the role of blockchain technology, enabling real-time transactions and financial activities, bringing up the creation of smart contracts that guarantee safe and efficient logistics-related transactions. The Implementation and Development section outlines the approach, milestones, and release dates of the system, such that a road map is laid out for further development and deployment. The key deliverables of this work include the representation of integrated systems for logistics optimization, the possibility of drastic reductions in costs, and improvement in operational efficiency through intelligent resource management. The deliverable provides a detailed discussion on technical design, component integration and conformity with the specifications of the project, thus being a comprehensive document for stakeholders involved in logistics and technology implementation.

Table of Contents

1	Executive Summary	5
	Table of Contents	6
	Table of Figures	8
	Table of Tables.....	9
	Definitions, Acronyms and Abbreviations.....	10
2	Introduction	11
	2.1 Objectives of the Deliverable	11
	2.2 Structure of the Document	11
3	Synchromodal Operations and Shared Resources	12
	3.1 Definition and Scope	12
	3.2 Benefits and Impact	12
	3.3 Integration with Existing Logistics Frameworks	13
	3.4 Components Analysis.....	14
	3.4.1 Resource Monitoring and Events Manager (REM).....	15
	3.4.2 Monitoring Module (MOM)	18
	3.4.3 Event Management Module (EMM)	19
	3.4.4 Fleet Monitoring Manager (FMM)	21
	3.4.5 Mitigation Manager (MIM)	21
	3.5 Alignment with Project Architecture and Requirements.....	25
	3.5.1 Project Architecture	25
	3.5.2 Requirements for REM	26
4	Intelligent Services for Logistics Optimization	29
	4.1 Planning Scheduling and Routing.....	30
	4.1.1 Middle Mile Logistics.....	30

4.1.2	First and Last Mile Logistics.....	39
4.1.3	Last Mile Logistics with Marsupial Systems	49
4.2	Optimal Path Finding Algorithms	53
4.2.1	Ground Vehicles	53
4.2.2	Aerial Vehicles	55
4.3	Routing and Scheduling Services in the TRACE System	60
4.3.1	Integration with Existing Logistics Frameworks	61
4.3.2	Alignment with Project Architecture and Requirements	61
5	Blockchain Infrastructure for Smart Contracts.....	66
5.1	Overview of Blockchain Technology.....	66
5.1.1	Technical Overview of Algorand.....	68
5.2	Smart Contracts for Real-Time Transactions	72
5.3	Financial Operations and Authentication	76
5.4	Alignment with Project Architecture and Requirements	93
5.4.1	Self Sovereign Identity (SSID) Module: (Blockchain Based Authentication System)	94
5.4.2	dNFT Module.....	100
5.4.3	Next Steps and Future Roadmap	104
6	Implementation and Development.....	107
6.1	Methodology.....	107
6.2	Phases and Milestones.....	108
6.3	Release Plan (M18 and Subsequent Releases).....	108
7	Conclusion.....	110
8	References.....	111

Table of Figures

Figure 1: Architecture of REM and interaction with other components	14
Figure 2: REM Architecture	15
Figure 3: TRACE Event Sequence Diagram	24
Figure 4: REM in TRACE Architecture	26
Figure 5: Middle Mile Logistics with the option of a 3PL involvement	31
Figure 6: Vehicles from set N are dispatched to retrieve shipments from vehicles in set M and finish the delivery.....	37
Figure 7: Indicative Cost Matrix for Route Optimization Problems	42
Figure 8:A graph representation of the intermediate and the delivery points.....	50
Figure 9: An example of a change in the shortest path after excluding the polygonal area.....	54
Figure 10: Geo-fenced zone with no-go-zones inside for the route generation of the UAVs	56
Figure 11: Vertical take-off and landing for UAVs, with a transition among positions in a user-defined constant altitude	57
Figure 12: Trajectory examples in free space and geo-fenced zones	59
Figure 13: T4.4 in TRACE Architecture	60
Figure 14: The flow diagram of the Scheduler’s and Route Optimizer’s functions during a request.	62
Figure 15: The Scheduler and Route Optimizer with their internal architecture.....	64
Figure 16: An overview of transaction flow in Algorand.....	72
Figure 17: Decision-making process for blockchain consensus of the TRACE system's Blockchain module	75
Figure 18: SSID Registration	95
Figure 19: SSID Login.....	100
Figure 20: DNFT Module	104

Table of Tables

Table 1: Vehicle specifications for the simulated testing example.....	46
Table 2: Shipment details for the simulated testing example.....	47
Table 3: Overall summary for total distance minimization	47
Table 4: Detailed route results for total distance minimization.....	47
Table 5: Overall summary for total time minimization	48
Table 6: Detailed route results for total time minimization	48

Definitions, Acronyms and Abbreviations

Abbreviation	Definition
3PL	Third-Party Logistics Provider
API	Application Programming Interface
dNFT	Dynamic Non-Fungible Token
EMM	Event Management Module
FMM	Fleet Management Module
MIM	Mitigation Manager Module
MOM	Monitoring Module
PKI	Public Key Infrastructure
REM	Resource and Event Management
SDK	Software Development Kit
SSID	Self Sovereign Identity
UAV	Unmanned Aerial Vehicle
UCV	Urban Consolidation Center
UGC	Unmanned Ground Vehicle
VRP	Vehicle Routing Problem

2 Introduction

2.1 Objectives of the Deliverable

The objective of this deliverable is to present and analyze the synchromodal operations and optimization of shared resources of the event management framework. Initially, this report focuses into the details of the event management operations, taking into consideration the technological capabilities and the limitations of infrastructural interfaces, and the Streamhandler enabler along with user requirements. More specifically, this deliverable emphasizes on the development of algorithms, mechanisms and processes to systematically examine the design space of synchromodal logistics. TRACE directly targeting deployment aspects of physical (infrastructure level) and overlay (service level) topology, placement, and overall system dimensioning, including aspects of resilience and availability. Also, in this deliverable the essential functionalities for the events management large networks logistics will be implemented, and the proper functions to collect, analyze, process and integrate such events in the TRACE platform will be designed. The task will feed the logistics operations and intelligent scheduling (T4.4), the business opportunities and modelling (T5.3) and the social innovation and good practices (T5.4) with crucial input material on the exact technological mixture under investigation, with a particular focus on the above deployment aspects.

2.2 Structure of the Document

This deliverable is divided into three sections to provide a comprehensive overview of the synchromodal operations and optimization of the shared resources of the event management framework. Synchromodal Operations and Shared Resources , analyzes the role and abilities of synchromodal operations that are adopted in the TRACE platform to deliver the optimal transfer plan for the benefit of logistic companies. In the Intelligent Services for Logistics Optimization processing capabilities to optimize logistics and enable intelligent scheduling, leveraging real-time data on cargo, speed, position, and delivery goals, while incorporating factors like environmental impact, traffic conditions, delivery prioritization, and platoon capacity to achieve minimal delivery times. Blockchain Infrastructure for Smart Contracts introduces the way that the Blockchain establishes trust between logistics partners in the TRACE project. In the section Implementation and development of all the components are discussed.

3 Synchromodal Operations and Shared Resources

3.1 Definition and Scope

Synchromodal operations involve the possibility of a logistic system to dynamically switch between different transporting modes [1] in real time, depending on operational conditions, customer demands, and resource availability. Synchromodal coordination can also be seen as a dynamic, multimodal way of choosing cargo and transport modes - such as trucks, rail, or rambles - in real time according to changing circumstances, to ensure the most productive and ecologically-friendly outcome. The purpose of this view will enable coordination frameworks to easily select various options of transport modes concerning operational conditions, customer demand, and resource availability, so as to achieve maximum flexibility and efficiency with environmental sustainability. TRACE provides solutions on intelligent monitoring and an event management system for real-time decision-making, smart re-scheduling, and the creation of smart contracts to guarantee transparency. Key objectives concern resource sharing across multiple modes of transport, the better utilization of vehicles, reduction of energy use by using less vehicles and financial operational costs compared to the current status. In addition, the deliverable will be in full compliance with the remaining general objectives of the project, such as logistics chains' flexibility and the use of digital technologies, like blockchain, with the ability to conduct secure, transparent operations. These synchromodal operations are being realized within the TRACE platform through the coordination of intelligent decision-making algorithms that allow the transportation of resources across various activities for the benefit of companies. By exploiting this real-time information obtained by IoT sensors, location units, and cloud-based preparation frameworks, extend points are empowered to allow energetic mode-switching across truck, rail, and autonomous vehicles transport modes. This ensures that delays are constrained, carbon dioxide emission is reduced, and the logistics services assurances are reliable under variable operational conditions.

3.2 Benefits and Impact

The synchromodal operations through TRACE are envisioned to realize significant benefits through a logistic ecosystem for operational efficiency and sustainability. Key advantages include the possibility of real-time adaptability to disruptions such as congestion, weather events, or infrastructure issues that typically build inefficiencies. For example, when a truck is broken down TRACE platform will be used to identify the closest trucks to it in order to complete the delivery of high priority shipments in the destination. These operations monitor such conditions in a dynamic way and apply AI/ML-driven decision-

making systems that amplify responsiveness and flexibility, ensuring seamless transitions using different carriers or modalities.

Such a system will ensure that resources - which may lower fuel consumption and, consequently, reduce emissions and operational costs. For instance, in the case of truck delivery scheduled in the system, if it reaches some point where it hits traffic, then it could automatically re-route that truck, or even transfer the cargo to rail if one of those alternatives were most efficient at that specific point in time. The flexibility ensuing would hence lead to economies in cost, less impact on the environment, and an enhanced level of delivery service.

Among the key benefits are the unification of multiple data sources, such as traffic information, the continuous monitoring of on-board sensors of a vehicle, and weather conditions, on a single platform. This basically gives the assurance that in logistics decisions, data-driven and predictive insights are being considered, capable of foreseeing disruptions well in advance and offering pre-emptive resource reallocation. TRACE aims at operational flexibility within the logistics networks, where one can easily switch in real time from one mode of transport to the other. It will therefore improve resource utilization, cost efficiency, and energy consumption. For example, if a truck operates routes to a specific destination and it has additional space, then it can be used to transfer shipments from other companies to the destination. In addition, the synchromodal framework supports sustainability goals in how it optimizes transportation modes in real time to reduce overall environmental footprint from logistics operations to be able to contribute to the bigger project objective, which is lowering Green House Gas emissions. For example, if a train operates routes to a specific destination and it has additional space, then it can be used to transfer shipments from other companies to the destination decreasing the number of vehicles making the transportation process more environmental friendly.

3.3 Integration with Existing Logistics Frameworks

Those operations play a very important role in integrating synchromodal logistics solutions with the existing logistics frameworks and technologies. A key aspect of such integration pertains seamless communication across MOM, EMM and FMM operating in a cloud environment. These are to be designed to be interoperable with any existing logistics system by using REST APIS or GUIs to retrieve the appropriate information from the logistic systems. TRACE will also be able to support the monitoring of logistics transfers. The synchromodal solution allows the blockchain technology to check if the supply chains are

verifiable and traceable. For example, in case that the EMM detects an event and consequently there is a change in delivery, blockchain-enabled notification of this information in a secure way to all freight operators, logistics managers, and end customers creates trust through transparency in the logistics transaction. The TRACE sychromodal solution will be integrated into existing logistics frameworks through active collaboration with the industry stakeholders and partners participating in pilot trials. The following figure shows in detail the communication and high-level architecture of REM and the modules it contains as well as its communication with the Streamhandler and the Scheduler.

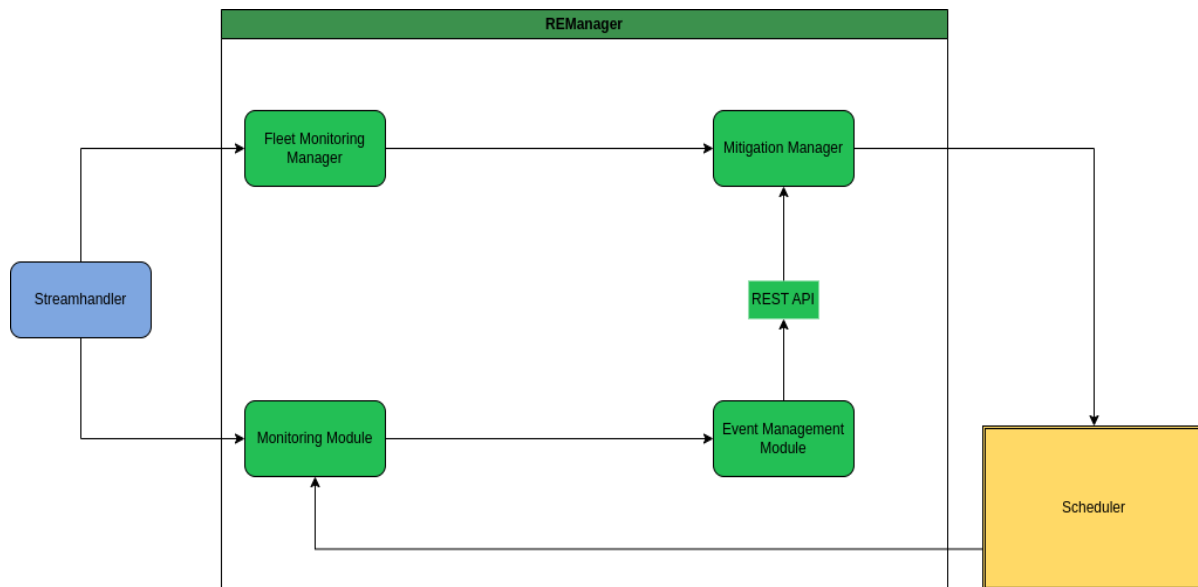


Figure 1: Architecture of REM and interaction with other components

The TRACE platform will be implemented in such a way in order to guarantee the perfect compatibility of sychromodal operations with today's logistics workflows and management systems. In this way, logistic companies will easily adopt sychromodal solutions without major infrastructural changes. TRACE will therefore adopt standard communication protocols and APIs that guarantee seamless data exchange between TRACE and existing systems for interoperability, wide-scale adoption of sychromodal operations, enhancement of the general logistics infrastructure, while at the same time allowing the sector to shift seamlessly into more flexible and efficient operations.

3.4 Components Analysis

The core component is the REM is designed to facilitate real-time logistics optimization and efficient event management through seamless data exchange and interaction between its components.

Each module contained in the REM used to monitoring, decision-making, and mitigation processes to ensure dynamic adaptability in freight operations. Following the REM architecture.

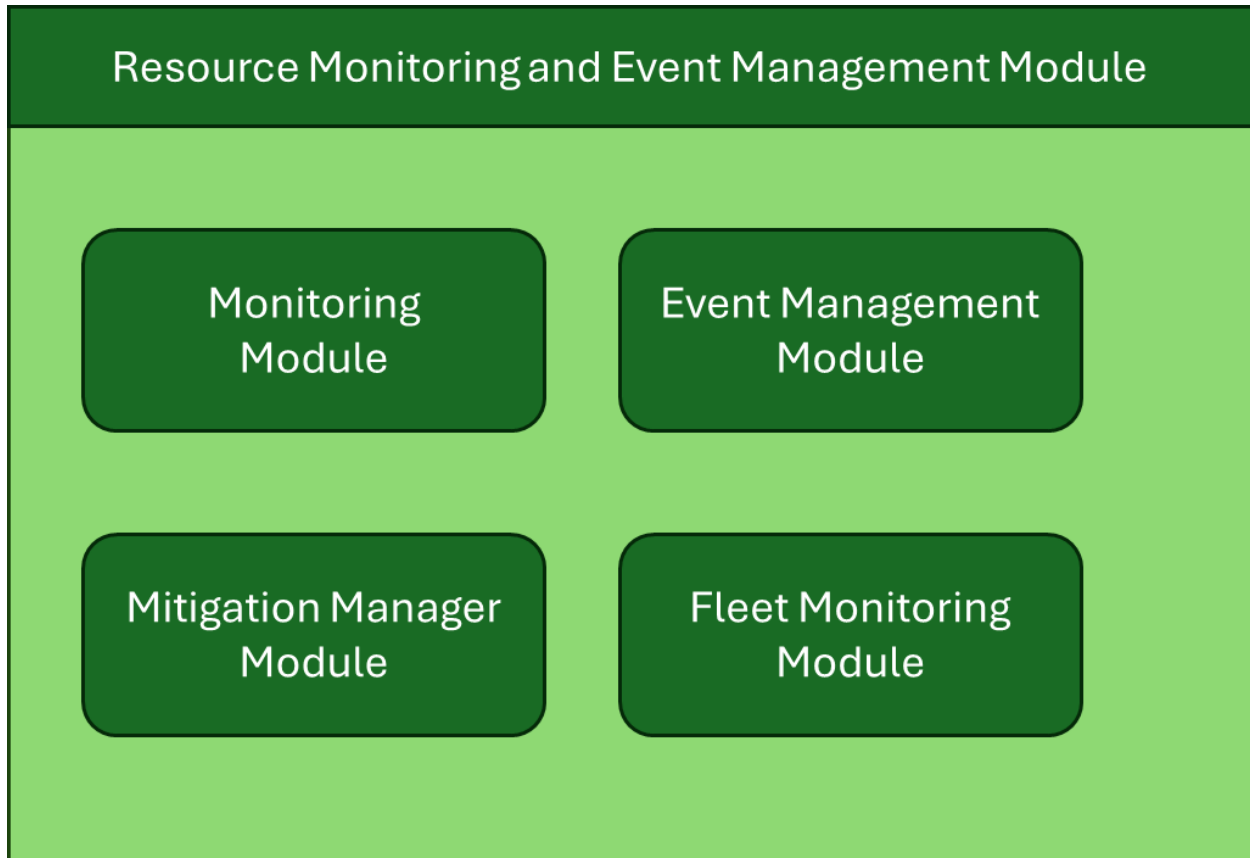


Figure 2: REM Architecture

3.4.1 Resource Monitoring and Events Manager (REM)

The REM is an essential component of the core layer within the synchromodal logistics framework. It acts as the coordinator in real-time resource monitoring and event-driven logistics decision-making. On the cloud environment level, it interacts with the other system components, in a main stream-handling base, to optimize the transportation routes and the allocations of resources. The following sections present a closer look into its core functions, interactions, and technological components.

Core Functions

1. **Data Monitoring:** Real-time information comes from multiple sensors concerning the status of a vehicle, traffic, and current environmental conditions. In fact, all this real-time data gives considerable information to the system for assessing the present status of logistics operations.
2. **Event Detection:** The REM shall interact with the EMM in the detection and processing of events that may occur to affect the effectiveness of logistics operations. It is designed to detect a wide range of events to ensure proactive interventions. These events include scenarios such as vehicles being stopped with the engine check alert activated, high traffic congestion triggering route reallocation, unexpected delays, blocked roads, adverse weather conditions disrupting operations, or hardware failures.
3. **Triggering the Scheduler:** REM identifies that there is new data or relevant events present, in turn triggering the Scheduler for reassignment of resources or diversion of deliveries. By recalculation, the Scheduler finds the most feasible transport mode or routes under disrupted conditions. For example, if a truck encounters a traffic holdup, the Scheduler is triggered to find an alternative route in order to circumvent the part of the initial route where the traffic is heavy. In case the initial route of the truck is not accessible anymore due to various reasons (e.g. weather conditions, accident, etc.) a re-scheduling and change of mode can be realized. By identifying such events, the REM ensures timely decision-making, triggering the Scheduler to reallocate resources or apply new routes as necessary, minimizing disruptions and maintaining operational efficiency.

Technological Infrastructure

REM works in a cloud environment that ensures scalability and high availability. The architecture of the cloud supports this for handling huge volumes of streaming data from distributed IoT sensors emanating from various transport modes.

- **Interaction Chain:** REM represents a part of an integrated process flow, starting right from the collection of data, to route adjustments.
- **StreamHandler:** Collects sensor data or event updates and streams the respective components of the TRACE platform e.g. the MOM, FMM etc.

All this information is fed into the REM, which interprets these events process the data and reassesses the current allocation of resources. Then, based on possible unexpected events, it triggers the Scheduler to compute new transportation routes or reallocations, if necessary.

Communication and Integration

- **Inter-module Communication:** REM communicates continuously with the EMM, Scheduler and MOM. In this regard, the system acts on real-time data and event triggers when there is immediate logistical adjustments that would need to be performed. It interacts with these modules through APIs and/or standard protocols.
- **Seamless integration with the synchromodal framework:** REM will be integrated into the TRACE platform and allow total interoperability with existing systems. REM uses cloud-based APIs and communication protocols to exchange data with these systems, hence offering to logistics operators and carriers the potential of upgrading their operations with dynamic routing and resource optimization capabilities without essentially overhauling the infrastructure at hand.

Impact on Synchromodal Operations

- **REM:** Will support real-time decision-making by continuous monitoring of sensor data and events for which it reacts when necessary by invoking the appropriate component (e.g., MOM for the monitoring of data and EMM for the detection of events). In such a way, synchromodal operations can dynamically respond to disruptions resulting in less delay, better usage of vehicles, and efficient use of resources.
- **Improved Efficiency and Sustainability:** The core mandate of the REM is to optimize resource utilization by efficiently addressing unexpected events based on updated and real-time data. In this respect, it might route a truck through a path with less congestion or change its mode of transport to something more energy-efficient upon disruption detection.
- **Operational Flexibility:** This REM contributes to operational flexibility for the logistics network. Since the REM continuously monitors the resources and triggers Scheduler in case of events, it thus provides dynamic response of the system to changes that might occur within the operational environment.

3.4.2 Monitoring Module (MOM)

The MOM is an important module in the REM. It performs in real time the collection of data, and the checking of possible deviations with the pre-defined plans for the routes. The module certainly plays an important role in ensuring the integrity and efficiency of logistical processes through constant analysis of sensory and operational data emanating from integrated assets. The functionality, interaction, and overall importance of the system analyzed below.

Core Functions

1. **Data Collection:** MOM directly connects to the StreamHandler, feeding sensory and other relevant data in real time continuously into the system. Such streams of data are several logistics parameters like location of vehicle, fuel consumption, traffic flow condition, environmental factors, and real-time update of transportation mode. MOM collects the data and arranges them properly in real time to monitor execution of logistics plans.
2. **Deviation Detection on transfer route plans:** The MOM is also supposed to monitor deviations from the transfer plans laid down by the Scheduler and the Route Optimizer. The Scheduler defines what routes are best and what allocations are to be done, and the MOM sees to it that the execution follows this plan. If deviations happen by way of constructions on the roads that are included in pre-defined routes, unexpected traffic, and other issues related to keeping off planned routes, the MOM detects such issues and flags them for corrective action.

In case of deviations, the MOM automatically triggers the EMM. The latter decides on the action needed given the deviation and verifies if logistics flow needs an adjustment. Further support is ensured through seamless integration, enabling immediate responses to disruptions and maintaining operational effectiveness with minimal impact on the overall logistics schedule

Technological Infrastructure

- **Cloud-based architecture:** Similar to other components within the synchromodal logistics framework, the MOM is designed with a scalable cloud environment so it could handle real-time processing of huge volumes of data. MOM, by using the cloud infrastructure, enables the monitoring of various logistics processes on different transport modes -such as trucks, rail, or drones- independently.

- **Integration with Core Components:** MOM works with several components of the logistical system, which are enumerated below:
 - **StreamHandler:** It provides the fundamental source of continuous data to MOM.
 - **EMM:** The detection of deviation requires the action of MOM, which communicates with the EMM to infer the corrective measure(s).

MOM interfaces, therefore, with the MIM in charge of mitigation and intervention processes and the Scheduler for online changes in routes and the reallocation of resources. Furthermore, the MOM will close the loop, updating its monitoring from the revised plans provided by the scheduler so that the system remains in step with operational realities.

3.4.3 Event Management Module (EMM)

The EMM is a module conceived to enhance decision-making in logistics. It covers value through the assessment of events detected during logistics operations. By using a rule-based system, the EMM analyzes the output provided by the MOM for an event that requires corrective actions or adjusting logistics plans. In the following section, the working, interaction, and significance of the module are elaborated upon.

Core Functions

1. **Event Evaluation:** The primary role of the EMM would be to assess the data provided by the MOM and determine events that may impact logistics operations. This shall include the assessment of deviations from intended routes, disturbances in service, or unexpected changes in conditions, such as traffic congestion or delays related to weather.
2. **Rule-Based Decision Making:** The EMM decides whether an event has occurred and if any intervention is called for using a rule-based approach. The system applies predefined rules in order to establish the relevance of the events detected, hence the module makes proper decisions on what changes should be done in the logistics operations. These rules may include thresholds of delays, tolerable levels of deviation, and conditions on which corrective actions become necessary. The rules have been defined after interaction with end-users to cover all the possible cases and scenarios that can arise during the logistics operations and simultaneously handled in the appropriate way detected events.

3. **Triggering corrective actions:** In the case of detecting a relevant event, the EMM communicates with the MIM in order to take an appropriate corrective action; this may update the logistics plans, readjust the set of resources used, or even reschedule the deliveries in order to maintain business operations with minimal disruption.

Technological Infrastructure

- **Architecture of EMM:** The module, EMM, should interact with the cloud to be capable handle massive volumes of data coming in from different sources, right to giving timely responses to logistics challenges.
- **Integration with Core Components:** The EMM interacts closely with several key components in the logistics system:
 - **MOM:** Continuous feeding from the monitoring of data and any deviation from planned operations. The MOM will feed the EMM to identify possible events that might be of interest and require evaluation or action.
 - **MIM:** Interact with the EMM in order to implement corrective actions identified from decisions based upon the detected event.
 - **Scheduler:** Transfer the updated logistics plans back to the Scheduler, thereby keeping logistics operations relevant in real-time.

Impact on Synchromodal Operations

1. **Increased Responsiveness:** The EMM immensely increases responsiveness in the logistics system. Also, analyzing real-time data coupled with rule-based decision-making aids in making interventions on time that can prevent minor disruptions from blowing out of proportion into major delays or a complete operational collapse.
2. **Event Management:** Since events can be categorized based on predefined rules, the EMM supports event management. It, thus, follows that informed decisions can be taken in the minimum time by logistics operators to achieve an optimal allocation of resources and service levels, even in response to unforeseen circumstances.
3. **Operational Efficiency:** The EMM ensures overall efficiency in the logistics network through proper deviation management from planned operations. It will maintain this at an optimum flow by making necessary adjustments with speed and minimizing the impacts of disruption on delivery schedules.

3.4.4 Fleet Monitoring Manager (FMM)

Core Functions

1. **Data Collection:** The FMM interfaces directly with the StreamHandler, which continuously transmits various vehicle data in real time to the system. Such information streams encompass various logistics parameters, including vehicle location, availability, and capacity. The FMM gathers and organizes data in real time to oversee the positioning and availability of the fleets.
2. **Range-Based Decision Making:** The FMM monitors the availability and capacity of fleets within specified regions using a range-based methodology. The system uses established rules and logic to determine the relevance of the fleets provided; consequently, the module makes decisions based on information regarding which fleets are suitable for logistic operations. These regulations may encompass acceptable thresholds of deviation and conditions under which modifications are required.

Technological Infrastructure

- **Cloud-Based Architecture:** The FMM has been developed with a scalable cloud infrastructure to manage real-time processing of substantial data volumes. This architecture will enable the Fleet Monitoring Manager to efficiently process and regulate interactions among various components in logistics.
- **Integration with Key Components:** FMM works with several components of the logistical system, which are enumerated below:
 - **StreamHandler:** provides the fundamental source of continuous GPS data to the FMM.
 - **MIM:** the decision for relocation requires the action of the FMM, which communicates with the MIM to evaluate the availability and capacity of the fleet(s) within the specified region.
 - **MOM:** provides data related to the area of interest to the FMM.

3.4.5 Mitigation Manager (MIM)

The central role of the MIM is the implementation of the mitigation strategies regarding events detected by the EMM. This module represents a very important component in operations-continuity and

optimal exploitation of resources, adding efficiency in logistics operations. Functionality, interaction, and importance for the logistic ecosystem are analyzed in detail below.

Core Functions

1. **Implementation of mitigation plans:** Implementation of mitigation plans: Shall be the duty of the MIM to apply proper mitigation strategies that will have been decided. These will involve the analysis of the type of event detected and its consequences, so logistics operations can be maintained with minimal disruption. For example, in case that on a truck the EMM has detected a low level of fuel and a mitigation strategy that has been proposed is the stop in a gas station for re-fuelling then the consequences of this plan are analyzed. If this will cause an important delay then this mitigation plan will not be applied by MIM.”.
2. **Coordination with Scheduler:** One of the major tasks of the MIM will be to interact with the Scheduler for initiating re-allocations or adjustments to delivery plans. In case any event requires the reshuffling of resource allocation, the MIM shall contact the Scheduler regarding the shipment impacted and the vehicle available for prompt implementation of re-routing or re-assignments.
3. **Decision Making Flexibility:** MIM is capable of assessing whether a re-allocation should be performed concerning the context of the event detected. If the nature of the event does not have a significant impact on operations, the MIM may allow no action should be done. Hence the logistics processes are allowed to continue uninterrupted. For example, if for a truck the EMM detected a low level of fuel but there is enough to reach its destination then this is not a crucial event, so the vehicle continues without any action to be called by the MIM.

Technological Infrastructure

- **Cloud-Based Architecture:** The MIM operates at the cloud level to enable scalability, accessibility, and real-time decision capabilities. This architecture thus will allow the MIM to perform its work of processing effectively, controlling actions between different components.
- **Integration with Key Components:** The MIM interfaces with several other integral parts of this logistics system, namely:
 - **EMM:** It obtains input on the events detected and those that the EMM decides upon with respect to mitigation action.

- **Scheduler:** Coordinates with Scheduler for implementing the re-allocations or adjustments of the logistics plans because of event responses.

Impact on Synchromodal Operations

1. **Improved Utilization of Resources:** MIM provides for the efficient use of resources through timely reallocation and adjustment in logistics plans for the events detected. This ensures proper resource utilization to avoid delays and wastages within operation contexts.
2. **Continuity of Operations:** Through effective event response management, MIM continues to bear prime importance in ascertaining continuity within the logistics operations. It makes operations dynamically responsive to the changes in circumstances and, therefore, mitigates disruptions to the services being delivered.
3. **Improved Decision-making:** The MIM enhances decision-making in the logistical network. This is achieved by the decisions on reallocations and acting on recommendations through the EMM. These improve operational resilience at this level of adaptability.

The sequence diagram of Figure 2 analyses the workflow describing the interfaces between the modules.

A brief analyzation of the workflow is:

1. The Streamhandler starts by gathering sensor data and streaming it to the FMM
2. The FMM continuously streams data, providing real-time updates on vehicle statuses.
3. The MOM receives information from the FMM and sends the information for event detection.
4. The EMM may detect and report an event to the MIM if any.
5. FMM supplies available vehicle information to the MIM.
6. The MIM provides the appropriate mitigation plan based on the detected event and the available vehicles.
7. The MIM takes the received mitigation plan and executes it if required.
8. The MIM then sends any updated or new transfer plans back to the MOM.
9. The MOM detects any deviation from the transfer plans.
10. In case the deviation is found, then a new event is sent to the EMM for further actions.

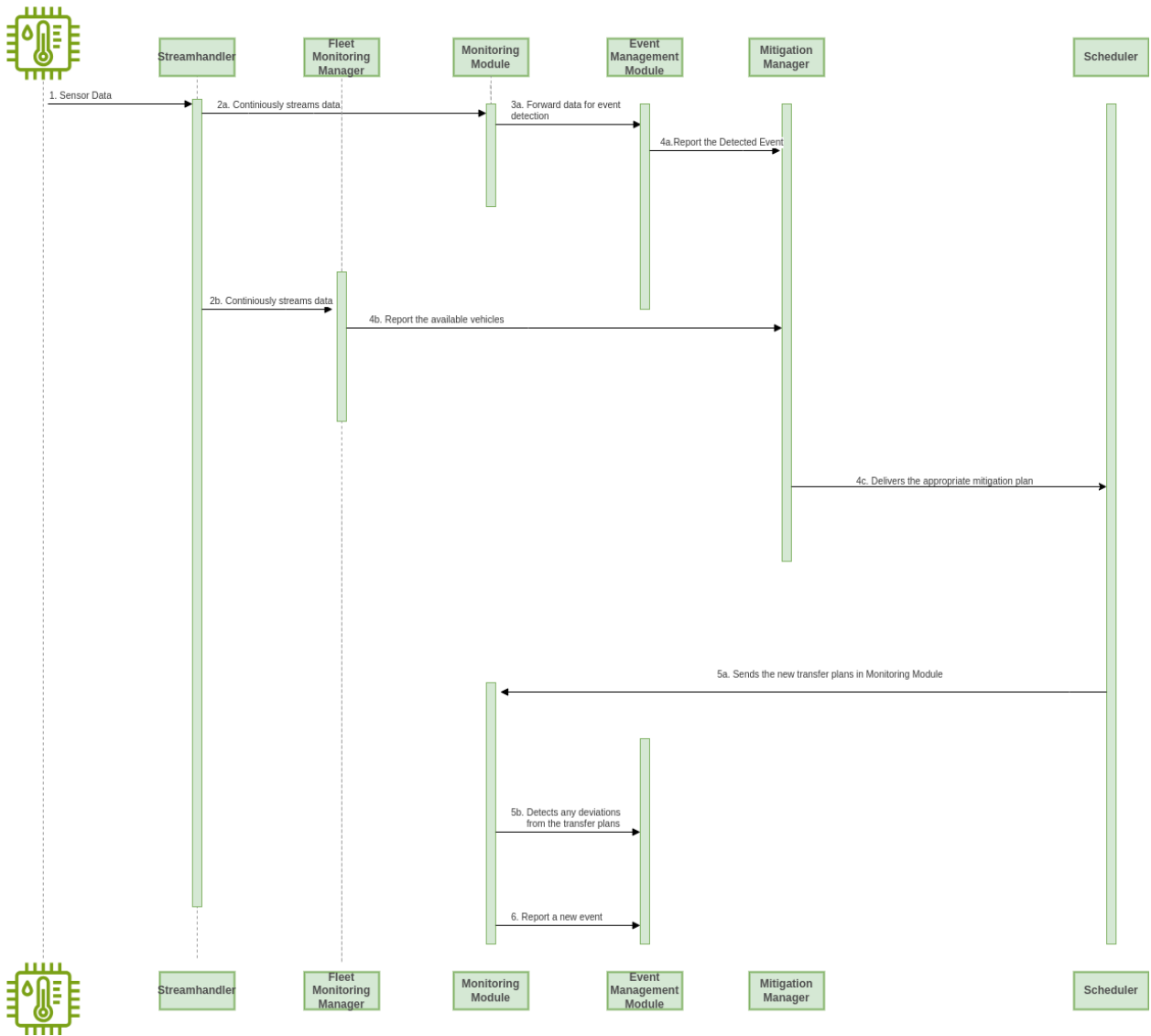


Figure 3: TRACE Event Sequence Diagram

3.5 Alignment with Project Architecture and Requirements

3.5.1 Project Architecture

The REM is the module closely linked with the overall architecture of the TRACE project, where it is analyzed extensively in the Deliverable 3.1, for facilitating seamless integration that maximizes operational effectiveness. Within the cloud-based framework of TRACE, REM resides at the core of aggregation points, which receive real-time data and identify events for resource optimization. It establishes a direct interface with the critical architectural components: the MOM, the EMM and the Scheduler, which ensures efficient communication responsive to disturbances. With its conformance to established communication standards and through interaction driven by APIs and the StreamHandler, the REM has promoted compatibility with pre-existing logistics frameworks. This orientation supports the REM in order to offer improvements on resource allocation across different transport modes, ensuring at the same time that this element serves effectively the goals of flexibility, scalability, and sustainability of the TRACE architecture regarding sychromodal logistic operations.

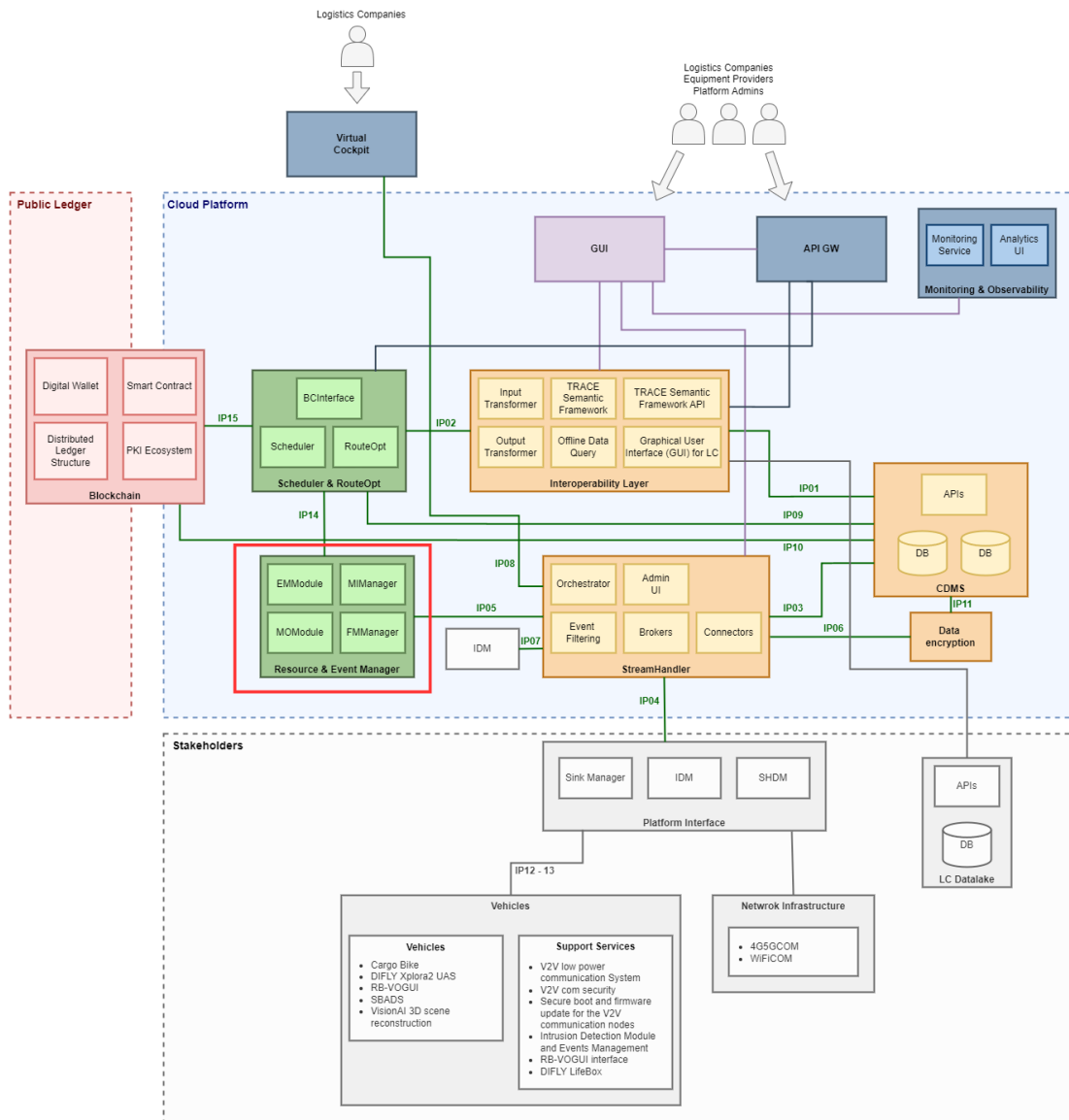


Figure 4: REM in TRACE Architecture

3.5.2 Requirements for REM

The Requirements that EMM satisfies:

- 1) EVT-FUN-001: Events should be generated when a) a shipment is loaded onto a vehicle, b) when the shipment is delivered to the platform or logistics company's system in real-time.
- 2) EVT-FUN-004: TRACE platform shall be capable of detecting various types of events, such as delays, disruptions, or anomalies, in real-time.
- 3) EVT-FUN-004: TRACE platform shall be capable of detecting various types of events, such as delays, disruptions, or anomalies, in real-time.

- 4) EVT-FUN-002: Event Management Module shall integrate with other Modules.
- 5) EVT-FUN-014: TRACE shall operating area limit fault event.
- 6) EVT-FUN-012: When an unmanned vehicle approaches the limits of its authorized operating area, TRACE shall trigger a warning event.
- 7) EVT-FUN-013: When an unmanned vehicle has a mechanical or electronic breakdown, TRACE shall trigger a fault event.
- 8) EVT-FUN-010: TRACE platform should use machine learning algorithms to identify patterns in event occurrences.
- 9) EVT-FUN-005: TRACE platform shall prioritize events based on predefined criteria to ensure appropriate response and resource allocation.

The Requirements that FMM satisfies:

- 1) EVT-FUN-012: When an unmanned vehicle approaches the limits of its authorized operating area, TRACE shall trigger a warning event.

The Requirements that MIM satisfies:

- 1) EVT-FUN-006: TRACE platform shall provide notification and alerting capabilities to inform users about important events, updates, or changes in transportation operations, ensuring timely awareness and response.
- 2) EVT-PRM-001: TRACE platform shall support automated responses to certain types of events, such as triggering re-allocations or updating schedules.
- 3) EVT-FUN-011: When a low-battery alert event is fired from an unmanned vehicle, TRACE should be forced to return to consolidation center for a battery charging or switching.
- 4) EVT-FUN-003: TRACE platform may allow for the assessment of the impact of events on various aspects of logistic processes, including operational costs, time delays, and customer satisfaction.
- 5) EVT-FUN-007: TRACE platform shall store actions taken in response to specific events (e.g., assessing the effectiveness of the action and the timeframe for its execution). This confirms the suitability of each action for a particular event, enabling the platform's machine learning capabilities to improve future actions for similar events.

The Requirements that MOM satisfies:

- 1) EVT-FUN-005: TRACE platform shall prioritize events based on predefined criteria to ensure appropriate response and resource allocation.
- 2) MON-PRM-001: TRACE platform may support scenario modeling and what-if analysis to evaluate the impact of different operational strategies, route optimizations, or resource allocations on performance metrics.

4 Intelligent Services for Logistics Optimization

Intelligent Services for Logistics Optimization are advancing pick-up and delivery operations by transforming traditional methods into optimized, technology-driven systems. In non-optimized logistics processes, although current systems include some functionalities, such as tracking shipments' due dates, the human factor still plays a significant role. To a great extent, this happens because such systems often provide limited – to none - scheduling capabilities for complex multi-stop deliveries, handling of specific time-windows, or real-time adjustments to unforeseen disruptions. As a result, manual interventions are frequently required, which can lead to inefficiencies, delays, and higher operational costs, ultimately impacting customer satisfaction and overall efficiency. In contrast, optimized delivery systems consider multiple parameters and constraints to provide optimal schedules and routes according to identified key performance indices.

In first-mile or last-mile deliveries, vehicles begin their route at a depot, where they either collect goods to be transported to various destinations or deliver goods to a set of predefined locations. The efficiency of these initial and final stages of the supply chain influences the overall logistics performance, cost, and delivery times. The underlying problem in the first-mile and last-mile deliveries is the well-studied **Vehicle Routing Problem (VRP)** [2, 3, 4, 5]. The single depot VRP posed as an optimization problem consists of a set of vehicles and a weighted graph, with one of its vertices denoted as the depot, while the rest being delivery destinations. The weight of the graph's edges corresponds to the cost of travelling from one vertex to another according to the key performance indices. The optimization goal is to find routes for the vehicles to visit all the delivery destinations that minimize the total cost. Acquiring solutions to this problem involves algorithmic techniques such as mixed integer linear programming, heuristics, machine learning, and metaheuristic algorithms [3, 6, 7, 8, 9].

By integrating intelligent services and combining conventional and autonomous vehicles, companies can significantly enhance first-mile and last-mile operations. Intelligent services enable precise route planning and adaptive responses to last-minute changes due to disruptive events, resulting in faster, more efficient, and often more environmentally sustainable deliveries. These delivery solutions provide a competitive advantage that results in an automated and, therefore, more robust system. In the TRACE project, in the context of T4.4, a set of optimization solutions regarding the whole spectrum of the transportation chain have been implemented and will be tested in the second reporting period of the project. Specifically, these solutions include a variety of techniques for first, middle, and last-mile logistics

optimization supporting heterogeneous manned and autonomous vehicles, marsupial systems and intelligent city logistics.

4.1 Planning Scheduling and Routing

The TRACE platform offers comprehensive support for each stage of the delivery process: the first mile, which covers the transportation of goods from the sender's location to the logistics company; the middle mile, which involves moving goods between depots within the logistics network; and the last mile, which focuses on the final delivery from the logistics provider to the recipient.

This section outlines the methodologies used to provide optimal scheduling and routing solutions within the platform. To begin with, to find an optimal schedule and routes for the vehicles, appropriate versions of the VRP are addressed. However, finding an optimal route entail knowing the cost of traveling from one address to another, this issue will be addressed in Section 4.2. In this section, we discuss how the platform offers services to middle-mile delivery, even in the case where an event disrupts the transportation of goods, first-mile and last-mile delivery scenarios. Finally, the last-mile delivery in a marsupial environment is discussed separately, that is, when marsupial vehicles are involved. Marsupial vehicles are vehicles capable of deploying or retrieving other vehicles (i.e., a carrier vehicle) or being deployed or retrieved (i.e., a passenger vehicle).

4.1.1 Middle Mile Logistics

In logistics, *Middle-Mile Delivery* refers to the stage of transportation that occurs between the initial departure from a supplier or warehouse and the final delivery to a retail location, distribution center, or sorting facility¹. The users are logistics companies that need to transfer shipments 3PL), as Figure 1 presents

The user provides a set R of shipments where every $r \in R$ has:

- weight $w_r \in R^+$,
- priority $q_r \in N$ (the lower the number the higher the priority),
- volume $v_r \in R^+$,

¹Donald, J. Bowersox, David, J. Closs, M Bixby, Cooper, John, C. Bowersox, “**Supply chain logistics management**”, 2020, McGraw-Hill Education

- a deadline $t_r^{(d)}$ to be at its destination

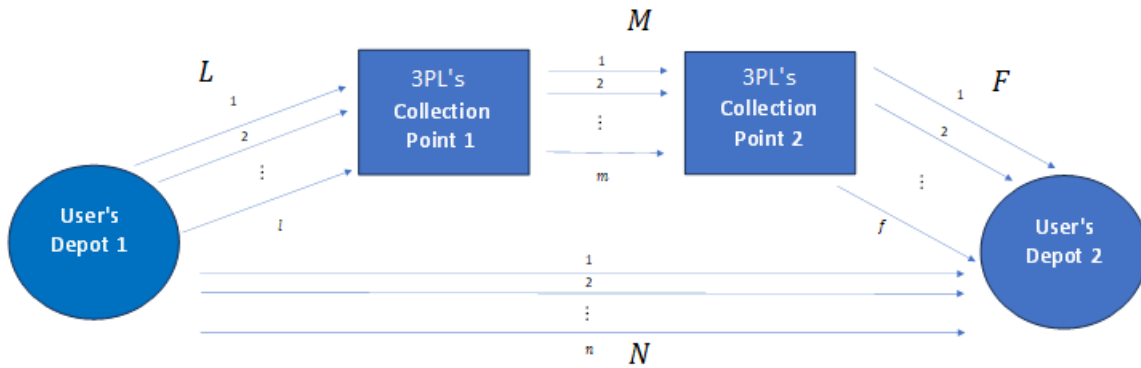


Figure 5: Middle Mile Logistics with the option of a 3PL involvement

The user must transfer shipments from “User's Depot 1” to “User's Depot 2” either by employing a third-party logistics company or using owned trucks. L, F, ML, F, M , and NN are the sets of available vehicles or scheduled trips for every leg of the transportation, where $|M| = m, |L| = l, |F| = f$ and $|N| = n|M| = m, |L| = l, |F| = f$ and $|N| = n$ are their cardinalities, i.e. the numbers of the available vehicles.

The third-party logistics companies provide:

- a set of vehicles M ,
- a carrying unit price $p_j^{(car)} \in R^+, j \in M$, per weight of shipment
- their schedule $S = \left\{ \left(t_j^{(dep)}, t_j^{(arr)} \right) \subseteq [0, +\infty) \times (0, +\infty) : j \in M \right\}$, with the departure, and arrival times.

The user provides the minimum and the maximum number of vehicles, $l_{min}, l_{max} \in N$, that could make direct deliveries without a company. The user has access to:

- a set L of available vehicles for transportation from the user's starting depot to the LC's collection point 1 (from User's depot 1 to 3PL Collection Point 1, see Figure 4).
- a set F of available vehicles for transportation from the 3PL's collection point 2 to the user's destination depot (from LC Collection Point 2 to User's depot 2 in Figure 4).
- a set N of available vehicles for transportation from the user's starting depot to the user's destination depot (from User's depot 1 to User's depot 2 in Figure 4).

Each mode of transport has a capacity W_j and V_j , $j \in L \cup M \cup F \cup N$, for the weight and volume, respectively. Furthermore, each mode of transport contains at most $N_j \in N$ shipments, $j \in L \cup M \cup F \cup N$. Every set of transports, consequently every vehicle, has to make a specified route with estimated distance d_j , travel time $t_j^{(travel)}$, loading and unloading times $t_j^{(load)}$ and $t_j^{(unload)}$, fuel price $p_j^{(fuel)}$, a unit operational cost per hour OC_j , a unit fuel consumption per distance FC_j^{empt} , when a vehicle travels without being loaded, and a fuel consumption per distance per weight FC_j^{ld} , $j \in L \cup F \cup N$.

By $\lambda_m, \lambda_q \in R$, we denote constants that balance the cost with valuing the early arrival of shipments with higher priority. Also, $\lambda_p \in R$ is the penalty cost of not being able to send a shipment. The objective is to find an allocation of the shipments to the vehicles that minimizes the delivery cost.

The mathematical modeling of the above problem is as follows.

Variables of the model:

$$y_{rj} = \begin{cases} 1, & \text{if the shipment } r \text{ travels with vehicle } j \in L \\ 0, & \text{otherwise} \end{cases}$$

$$x_{rj} = \begin{cases} 1, & \text{if the shipment } r \text{ travels with vehicle } j \in M \\ 0, & \text{otherwise} \end{cases}$$

$$z_{rj} = \begin{cases} 1, & \text{if the shipment } r \text{ travels with vehicle } j \in F \\ 0, & \text{otherwise} \end{cases}$$

$$u_{rj} = \begin{cases} 1, & \text{if the shipment } r \text{ travels with vehicle } j \in N \\ 0, & \text{otherwise} \end{cases}$$

$$\underline{y}_{rj} = \begin{cases} 1, & \text{if vehicle } j \in L \text{ makes a delivery} \\ 0, & \text{otherwise} \end{cases}, \underline{x}_{rj} \\ = \begin{cases} 1, & \text{if vehicle } j \in M \text{ makes a delivery} \\ 0, & \text{otherwise} \end{cases}$$

$$\underline{z}_{rj} = \begin{cases} 1, & \text{if vehicle } j \in F \text{ makes a delivery} \\ 0, & \text{otherwise} \end{cases}, \underline{u}_{rj} \\ = \begin{cases} 1, & \text{if vehicle } j \in N \text{ makes a delivery} \\ 0, & \text{otherwise} \end{cases}$$

Subject to the following Constraints:

The following constraints ensure route consistency:

Every shipment must travel with one vehicle or none

$$\sum_{j \in L} y_{rj} \leq 1, \quad \sum_{j \in M} x_{rj} \leq 1, \quad \sum_{j \in F} z_{rj} \leq 1, \quad \sum_{j \in N} u_{rj} \leq 1, \quad \text{for every } r \in R$$

Send a shipment either through the 3PL provider or the companies' vehicles

$$\sum_{j \in L} y_{rj} + \sum_{j \in N} u_{rj} \leq 1, \quad \text{for every } r \in R$$

Every shipment that travels through the 3PL provider reaches the destination

$$\sum_{j \in L} y_{rj} = \sum_{j \in M} x_{rj} = \sum_{j \in F} z_{rj}, \quad \text{for every } r \in R$$

Vehicle Consistency Constraints:

The following constraints ensure that if no shipments are assigned to a vehicle, then the corresponding variable is set to 0

$$\underline{y}_j \leq \sum_{r \in R} y_{rj}, \quad \text{for every } j \in L, \quad \underline{x}_j \leq \sum_{r \in R} x_{rj}, \quad \text{for every } j \in M$$

$$\underline{z}_j \leq \sum_{r \in R} z_{rj}, \quad \text{for every } j \in F, \quad \underline{u}_j \leq \sum_{r \in R} u_{rj}, \quad \text{for every } j \in N$$

The following constraints ensure that when at least one roller cage is assigned to a vehicle then the variable that corresponds to its selection is set to 1

$$\underline{y}_j \geq y_{rj}, \quad \text{for every } r \in R \text{ and } j \in L, \quad \underline{x}_j \geq x_{rj}, \quad \text{for every } r \in R \text{ and } j \in M$$

$$\underline{z}_j \geq z_{rj}, \quad \text{for every } r \in R \text{ and } j \in F, \quad \underline{u}_j \geq u_{rj}, \quad \text{for every } r \in R \text{ and } j \in N$$

This constraint is enforced when the vehicles of set L and N coincide and need to make a transportation to 3PL's collection point 1 or a direct transportation to User's depot 2

$$\underline{y}_j + \underline{u}_j \leq 1, \quad \text{for every } j \in L$$

The range of direct transfers:

This constraint ensures that the number of direct transfers is up to the user's needs

$$l_{min} \leq \sum_{j \in N} \underline{u}_j \leq l_{max}$$

Capacity Constraints:

$$\sum_{r \in R} y_{rj} \leq N_j, \quad \sum_{r \in R} y_{rj} w_r \leq W_j, \quad \sum_{r \in R} y_{rj} v_r \leq V_j, \quad \text{for every } j \in L$$

$$\sum_{r \in R} x_{rj} \leq N_j, \quad \sum_{r \in R} x_{rj} w_r \leq W_j, \quad \sum_{r \in R} x_{rj} v_r \leq V_j, \quad \text{for every } j \in M$$

$$\sum_{r \in R} z_{rj} \leq N_j, \quad \sum_{r \in R} z_{rj} w_r \leq W_j, \quad \sum_{r \in R} z_{rj} v_r \leq V_j, \quad \text{for every } j \in F$$

$$\sum_{r \in R} u_{rj} \leq N_j, \quad \sum_{r \in R} u_{rj} w_r \leq W_j, \quad \sum_{r \in R} u_{rj} v_r \leq V_j, \quad \text{for every } j \in N$$

Time Window Constraints:

For T a large enough constant

The shipments arrive at the user's depot 2 on time

$$\sum_{j \in M} x_{rj} (t_j^{(arr)} + t_j^{(unload)}) + \sum_{j \in F} z_{rj} \cdot t_j^{(travel)} \leq t_r^{(d)}, \quad \text{for every } r \in R$$

The shipments are loaded and delivered in time before the scheduled departure

$$\begin{aligned} t_{curr} + \sum_{j \in L} y_{rj} (t_j^{(travel)} + t_j^{(load)} + t_j^{(unload)}) &\leq \\ &\leq \sum_{j \in M} (x_{rj} (t_j^{(dep)} - t_j^{(load)})) + T \left(1 - \sum_{j \in M} x_{rj} \right), \quad \text{for every } r \in R \end{aligned}$$

Objective Function:

$$\begin{aligned}
 & \left(\lambda_m \sum_{j \in L} \left(d_j p_j^{(fuel)} FC_j^{ld} + t_j^{(travel)} OC_j \right) \sum_{r \in R} y_{rj} w_r + \lambda_m \sum_{j \in L} \left(d_j p_j^{(fuel)} FC_j^{empt} + t_j^{(travel)} OC_j \right) y_j \right. \\
 & + \lambda_m \sum_{j \in F} \left(d_j p_j^{(fuel)} FC_j^{ld} + t_j^{(travel)} OC_j \right) \sum_{r \in R} z_{rj} w_r + \lambda_m \sum_{j \in F} \left(d_j p_j^{(fuel)} FC_j^{empt} + t_j^{(travel)} OC_j \right) z_j \\
 & + \lambda_m \sum_{j \in N} \left(d_j p_j^{(fuel)} FC_j^{ld} + t_j^{(travel)} OC_j \right) \sum_{r \in R} u_{rj} w_r + \lambda_m \sum_{j \in N} \left(d_j p_j^{(fuel)} FC_j^{empt} + t_j^{(travel)} OC_j \right) u_j \\
 & + \lambda_m \sum_{j \in M} p_j^{(car)} \sum_{r \in R} x_{rj} + \lambda_q \sum_{r \in R} q_r \sum_{j \in M} t_j^{(arr)} x_{rj} + \lambda_q \sum_{r \in R} q_r \sum_{j \in F} t_j^{(travel)} z_{rj} \\
 & \left. + \lambda_p \sum_{r \in R} q_r \sum_{j \in M} \left(t_j^{(travel)} + t_{curr} \right) u_{rj} + \sum_{r \in R} \lambda_p \left(1 - \sum_{j \in L} y_{rj} - \sum_{j \in N} u_{rj} \right) \right)
 \end{aligned}$$

where $\sum_{j \in L} \left(d_j p_j^{(fuel)} FC_j^{ld} + t_j^{(travel)} OC_j \right) \sum_{r \in R} y_{rj} w_r$, corresponds to the cost of loading vehicle $j \in L$,

$\sum_{j \in L} \left(d_j p_j^{(fuel)} FC_j^{empt} + t_j^{(travel)} OC_j \right) y_j$ is the cost of the selected vehicles in L to travel empty.

The rest of the similar terms calculate the cost for the vehicles in the sets N and F .

$\sum_{j \in M} p_j^{(car)} \sum_{r \in R} x_{rj}$ is the cost of employing the vehicles of set M .

$\sum_{r \in R} q_r \sum_{j \in F} t_j^{(travel)} z_{rj} + \sum_{r \in R} q_r \sum_{j \in M} \left(t_j^{(travel)} + t_{curr} \right) u_{rj}$, is set to minimize the product of the priority with the arrival time of the shipment.

$\sum_{r \in R} \lambda_p \left(1 - \sum_{j \in L} y_{rj} - \sum_{j \in N} u_{rj} \right)$ calculates the penalty cost of not sending a shipment.

The output of the algorithm provides an assignment of the shipments to the vehicles with the corresponding cost.

Event Management – Vehicle breakdown

The optimization module for middle mile logistics operations also supports the handling of unexpected events that cause disruptions in the transportation procedure. In this scenario, we assume that an order has been placed from logistics company A to be transferred using a scheduled transportation from company B. During transportation a set of M vehicles of company B breaks down due to a

malfunction. Each vehicle in set M has a set of shipments $R_i, i \in M$, each shipment $RC_{ir}, i \in M$, and $r \in R_i$, has a corresponding weight $w_{ir} \in R^+$, volume $v_{ir} \in R^+$, and a priority $q_{ir} \in N$, where the smaller the number the higher the priority.

To address this event the user provides a set of vehicles N that are available to pick up the shipments that are sensitive. Each vehicle $j \in N$ has:

- a capacity W_j and $V_j, j \in N$, for the weight and volume, respectively
- a capacity of at most $N_j \in N$ shipments, $j \in N$
- fuel price $p_j^{(fuel)}, j \in N$
- a unit operational cost per hour $OC_j, j \in N$
- a unit fuel consumption per distance FC_j^{empt} , when a vehicle travels without being loaded, and a fuel consumption per distance per weight $FC_j^{ld}, j \in N$

The distances between the vehicles and the vehicle $d_{ji} \in R^+$, their estimated travel time $t_{ji}^{(travel)} \in N$, $(j, i) \in N \times M$, the distances between the broken vehicles and the logistics company's collection points $d_i^{(LCC)} \in R^+$, and their estimated times $t_i^{(LCC)} \in N, i \in M$, as presented in Figure 2, are considered known. By $\lambda_p \in R$ we denote the penalty cost of not transporting a roller that should have been transported. The objective is to match vehicles from set N to set M , which minimizes the cost of completing the delivery.

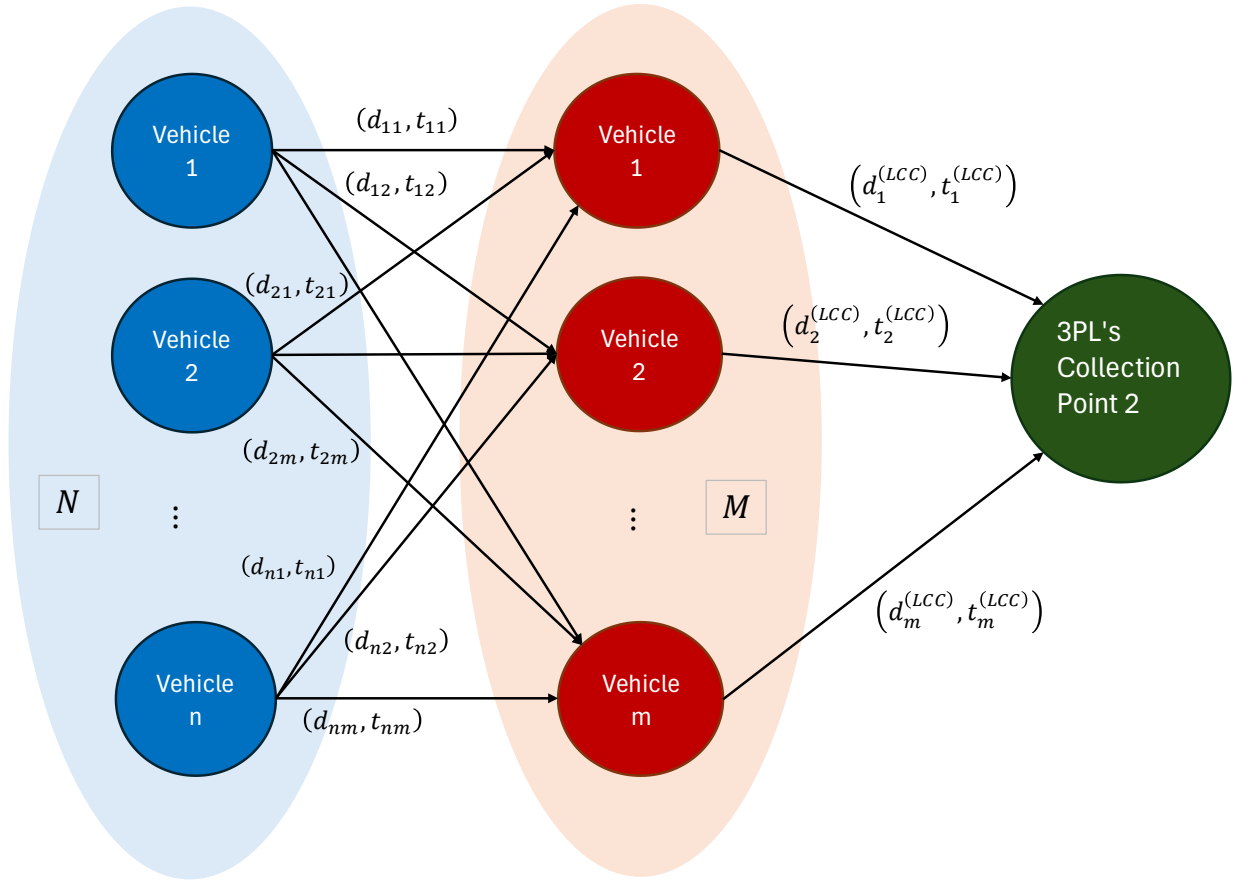


Figure 6: Vehicles from set N are dispatched to retrieve shipments from vehicles in set M and finish the delivery. The j -th vehicle, $j \in N$, picks up the cageshipments with the proper priority from the i -th vehicle, $i \in M$, that broke down by travelling distance d_{ji} and $d_i^{(LCC)}$.

The mathematical modeling of the above problem is as follows:

Variables:

$$y_{irj} = \begin{cases} 1, & \text{if the shipment } RC_{ir} \text{ is transported with vehicle } j \in N \\ 0, & \text{otherwise} \end{cases}$$

$$\underline{y}_{ij} = \begin{cases} 1, & \text{if the vehicle } j \in N \text{ picks up a shipment from vehicle } i \in M \\ 0, & \text{otherwise} \end{cases}$$

$$z_{ir} = \begin{cases} 1, & \text{if the shipment } RC_{ir}, (i, r) \in N \times R \text{ is selected to be transported} \\ 0, & \text{otherwise} \end{cases}$$

Subject to the following Constraints:

A roller cage is transported with to at most one vehicle:

$$\sum_{j \in N} y_{irj} \leq 1, \quad \text{for every } (i, r) \in M \times R_i$$

Matched Vehicles Constraints:

The following constraints ensure that if no shipments are assigned to a vehicle, then the corresponding variable is set to 0

$$\underline{y}_{ij} \leq \sum_{r \in R_i} y_{irj}, \quad \text{for every } (i, j) \in M \times N$$

The following constraints ensure that when at least one shipment is assigned to a vehicle then the variable that corresponds to its selection is set to 1

$$\underline{y}_{ij} \geq y_{irj}, \quad \text{for every } (i, r, j) \in M \times R_i \times N$$

Capacity Constraints:

$$\sum_{i \in M} \sum_{r \in R_i} y_{irj} \leq N_j, \quad \sum_{i \in M} \sum_{r \in R_i} y_{irj} w_{ir} \leq W_j, \quad \sum_{i \in M} \sum_{r \in R_i} y_{irj} v_{ir} \leq V_j,$$

for every $j \in N$

Objective Function:

$$\left(\sum_{i \in M} \sum_{j \in N} \left((d_i^{(LCC)} p_j^{(fuel)} FC_j^{ld} + t_i^{(LCC)} OC_j) \sum_{r \in R_i} y_{irj} w_{ir} \right) \right. \\ \left. + \sum_{i \in M} \sum_{j \in N} (d_{ji} p_j^{(fuel)} FC_j^{empt} + t_{ji}^{(travel)} OC_j) \underline{y}_{ij} \right. \\ \left. + \sum_{i \in M} \sum_{r \in R_i} \lambda_p \left(1 - \sum_{j \in N} y_{irj} \right) \right)$$

where $\sum_{i \in M} \sum_{j \in N} \left(\left(d_i^{(LCC)} p_j^{(fuel)} FC_j^{ld} + t_i^{(LCC)} OC_j \right) \sum_{r \in R_i} y_{irj} w_{ir} \right)$, corresponds to the cost of the loaded vehicle $j \in N$ to transport RC_{ir} to its destination,

$\sum_{i \in M} \sum_{j \in N} \left(d_{ji} p_j^{(fuel)} FC_j^{empt} + t_{ji}^{(travel)} OC_j \right) y_{ij}$ is the cost of vehicle $j \in N$ to travel empty to the location of vehicle $i \in M$.

$\sum_{i \in M} \sum_{r \in R_i} \lambda_p (1 - \sum_{j \in N} y_{irj})$ calculates the penalty cost of not sending a roller cage.

The output of the algorithm is an assignment of vehicles (a matching) to pick up shipments from the vehicles in M and deliver them to their destination.

For the implementation of the models Gurobi Optimizer² was utilized to provide an optimal solution.

4.1.2 First and Last Mile Logistics

While first-mile and last-mile logistics have distinct objectives and operations, they share similar operational frameworks and principles. Both involve the transportation of goods, often in urban environments, and the development of intelligent services for logistics optimization. As previously mentioned, these services are designed to increase efficiency by optimizing routes for maximum effectiveness and cost efficiency.

In first-mile logistics, operations begin at a central hub, for example, a consolidation center or depot, where vehicles are assigned to collect products from multiple manufacturers, suppliers, or warehouses in different locations. These vehicles travel various routes to collect shipments and transport them back to the hub. The main objective of the optimization tools, in this case, is to efficiently consolidate goods at the hub for further transportation to the next suitable hub through the middle mile, finally reaching their destination while managing capacity limits and optimizing key objectives. On the other hand, last-mile logistics also starts at the depot, but the goal shifts to deliver goods directly to various final destinations. In this case, vehicles leave the depot loaded with products that need to be distributed to different delivery points. Just like in first-mile operations, these vehicles must navigate routes efficiently, manage their loads, and optimize delivery schedules. After completing their deliveries, they return to the depot.

² <https://www.gurobi.com/solutions/gurobi-optimizer/>

Despite the differences in focus—collecting versus delivering—the underlying logistics operations of first-mile and last-mile logistics share many similarities. Each requires effective route planning, load management, and compliance with time constraints, making them closely connected in the logistics cycle. The TRACE pilot use-cases demonstrate the application of first-mile and last-mile logistics optimization solutions in urban environments. In Part C of the Greek pilot, autonomous vehicles, including UAVs and UGVs, execute last-mile deliveries to multiple customers inside the NKUA campus. Slovenian pilot Part A focuses on last-mile delivery, specifically the transfer of shipments from the mid-mile logistics to the interior of the BTC City area in Ljubljana. In Slovenian pilot Part B, a first-mile logistics provider delivers goods to the Urban Consolidation Centre (UCC) for consolidation before middle and last-mile dispatch. Finally, the Slovenian pilot Part C showcases a logistics company managing both first-mile and last-mile operations simultaneously, utilizing various transport methods. It is worth noting that while the Italian pilot also involves first-mile and last-mile logistics, it will be addressed separately due to its unique characteristics (extensively described in the following sub-section).

Solution Approach Overview

The successful design and implementation of a logistics optimization tool that efficiently handles real-world operations—especially in such complex scenarios involving multiple heterogeneous manned and unmanned vehicles, along with various logistics parties collaboratively performing transportation tasks—lies in the development of a model that accurately respects real-world parameters and limitations while translating them into constraints for the optimization process. Given that most vehicle routing and scheduling problems are NP-hard³, they are unlikely to be solvable in polynomial time. Therefore, addressing large-scale, real-world problems with numerous constraints and optimization variables can become extremely time-consuming or even infeasible. For this reason, the implementation of the logistics optimization tool within the context of the TRACE project employs the Google OR-Tools⁴ library—an open-source optimization tool designed for complex problems in vehicle routing, flows, integer and linear programming, and constraint programming. To handle the logistics challenge of resource allocation and optimal routing for a mixed fleet of heterogeneous vehicles for both operations, the solution developed and delivered in the context of T4.4 determines optimal.

³ Lenstra, J. K., & Kan, A. R. (1981). Complexity of vehicle routing and scheduling problems. *Networks*, 11(2), 221-227

⁴ Google OR Tools. Available online: <https://developers.google.com/optimization> (Accessed October 17, 2024)

Heterogeneous Fleet Management

In transportation operations, different vehicle types have unique characteristics that make them suited for specific delivery tasks. Some vehicles offer fast delivery speeds and can bypass ground traffic, making them ideal for urgent or short-range deliveries, but they often have limitations in payload capacity and range. On the other hand, some vehicles typically have a higher volume and weight capacity, making them more suitable for transporting bulkier shipments or accessing locations only reachable by the road network. By comparing the capabilities of various vehicles—including payload limits, delivery speeds, and operational range—the logistics model can effectively assign tasks that make use of each vehicle's strengths. This strategic assignment improves efficiency and ensures that operations are optimized for speed and capacity based on shipment requirements. Overall, utilizing a heterogeneous fleet enables logistics providers to optimize operations and improve service levels while maximizing resource use across the network. In the context of TRACE project, a variety of manned and unmanned vehicles, of different types, will be tested in cooperation during the pilot trials and demonstrations. This will evaluate their capabilities, benefits and weaknesses, both as individual vehicles and as part of a heterogeneous fleet.

Input Data and Configuration

The solver is designed with the capability to process a comprehensive set of input data that defines the parameters of various logistics problems. This includes detailed characteristics of each vehicle, such as payload capacity, speed, and operational range, which are essential for determining the most suitable vehicle for each delivery task. Additionally, specifics about the loads to be delivered, including weight, dimensions, and any special handling requirements, are taken into account. Geographic information about the delivery area is also crucial, including road networks and restrictions that may affect travel, such as geofencing and no-go zones. Geofencing allows the system to define virtual boundaries that restrict vehicle access to certain areas, while no-go zones indicate locations that vehicles must avoid altogether. Understanding these factors helps the solver create realistic and valid route layouts. Key inputs for optimization are the time and distance matrices, automatically generated from the initial data. These matrices detail travel time and distance between each pair of locations, forming the basis for calculating cost-effective route designs for each vehicle type. By using this data, the solver ensures that each assigned route is feasible and efficient, ultimately helping logistics providers minimize costs, improve delivery times, and maximize service levels.

Time-Distance Matrices

Time and distance matrices are essential for solving vehicle routing problems, as they provide critical data for effective route planning. Each matrix represents the travel times and distances between all pairs of locations, and they are customized for each vehicle type to reflect their unique characteristics, such as speed and operational range. More specifically, these matrices are structured as “ $n \times n$ ” grids, where “ n ” represents the number of locations that must be visited either for pickup or for delivery. Each cell in the grid captures the travel costs for every possible pickup and delivery point. The accurate representation of time and distance in these matrices enables the solver to perform complex calculations regarding total route costs, optimizing routes based on various criteria such as minimal travel time, shortest distance, or lowest operational costs. Figure 4 shows an indicative “cost” matrix for a route optimization problem, where H corresponds to the “home” location, D_i correspond to all the other locations that should be visited, and C_{i-j} corresponds to the cost value (the value that should be optimized), between the i -th and j -th locations.

	H	D1	D2	D3	D4	D5	...
H	0	C_{1-h}	C_{2-h}	C_{3-h}	.	.	.
D1	C_{1-h}	0	C_{2-1}				
D2	C_{2-h}	C_{2-1}	0				
D3	C_{3-h}			0			
D4	.				0		
D5	.					0	
...	.						0

Figure 7: Indicative Cost Matrix for Route Optimization Problems

Problem Constraints

Several operational constraints guide the model, organized to ensure feasible routing solutions. These include:

- **Capacity constraints:** Each vehicle has specific capacity limits based on *volume*, *weight*, and the *number of delivery lockers* it can accommodate. These constraints ensure that no vehicle is

overloaded, which could lead to delays or safety risks. Proper management of these limits is important for maintaining efficiency and complying with safety regulations.

- **Time windows and priority:** Deliveries are often restricted by specific time frames that indicate when items should be delivered. The model prioritizes urgent deliveries within these time frames to ensure customer satisfaction. For instance, if a delivery must be made within a specific time window, the model will optimize the route to ensure these deliveries take precedence over less time-sensitive ones.
- **Limited time constraints:** To ensure operational efficiency, total route time is restricted to remain within defined limits. This constraint considers factors such as vehicle battery life and charging requirements, ensuring that the vehicle can complete its assigned routes without running out of power. It is essential for planning routes since vehicles need to return to charging stations or stay within their range, ensuring timely deliveries and reducing delays.
- **Limited distance constraints:** Each vehicle has an operational range that cannot be exceeded. By ensuring that routes do not exceed the vehicle's maximum distance, the model improves reliability and reduces the likelihood of delays caused by battery issues or potential technical problems.
- **Vehicle-specific constraints:** Different types of vehicles have unique characteristics that may restrict their use for certain deliveries. For example, some vehicles may be unable to carry dangerous materials or fragile items. These constraints are critical for following safety regulations and operational policies. They ensure that each vehicle is utilized in a manner that aligns with its capabilities.
- **Pickup and delivery constraints:** The route optimization model incorporates specific pickup and delivery requests, allowing for pickups from various locations in addition to those from the depot. This approach aims to ensure that each delivery task corresponds to its designated pickup location. The planned constraints will require that a pickup must be completed prior to the associated delivery, with both tasks assigned to the same vehicle. Additionally, the constraints are designed to achieve effective load management throughout the entire pickup and delivery process, preventing any vehicle from exceeding its capacity.

At the time this deliverable is written, all the aforementioned constraints have been implemented and extensively tested in the logistics optimization module, except for the support of simultaneous pickups

and deliveries. This functionality is already in a mature stage of development; however, a stable, fully functional version that incorporates it is expected to be delivered in the following months of the project.

Objective Function

The objective function serves as a mathematical representation of the various costs associated with transporting goods from one location to another. It defines the transportation cost metric used to calculate the optimal solution for each delivery route. This cost can represent:

- **Actual cost:** This includes all expenses related to vehicle operation, such as vehicle and fuel costs. By incorporating these costs into the objective function, the model gives a realistic estimation of the financial impact of each route.
- **Total travel distance:** This metric involves summing all distances covered by the vehicles during the routing process. By optimizing routes to cover the shortest possible distance, organizations can reduce transportation expenses and promote environmental sustainability through reduced emissions.
- **Total travel time:** This refers to the cumulative time required for all vehicles to complete their assigned routes. By minimizing total travel time, the model improves delivery speed, which is vital for meeting customer expectations and service level agreements. Efficient time management can lead to increased customer satisfaction, and greater overall productivity.

The developed model allows users to select the transportation cost metric they wish to minimize. This flexibility allows for different transportation priorities and lets users adapt the solution to specific delivery needs.

Route Optimization Approach

With the optimization problem strictly defined, a two-phase algorithm facilitated by Google OR-Tools' range of heuristics and metaheuristics, identifies the optimal setup based on the problem definition while respecting the enforced constraints and satisfying the defined objectives. This process consists of two main phases: an initialization phase (**Phase 1**) and an improvement phase (**Phase 2**).

1. **Phase 1:** In the first phase of the optimization process, an initial solution is generated using a heuristic algorithm. This step is necessary because it establishes a feasible starting point for further optimization. While this initial solution may not be optimal, the choice of proper initial conditions for most optimization problems—and, thus, the heuristic used in this route

optimization scheme—can significantly affect both the feasibility and quality of the solution, as well as the overall execution time of the algorithm to find it. The key aspect of this phase is the dynamic selection of the most appropriate heuristic algorithm based on the specific characteristics of the problem under consideration. This adaptability ensures that the solution generated is both effective and possible, given the constraints and requirements of the scenario.

2. **Phase 2:** After generating an initial solution through heuristic algorithms in Phase 1, we proceed with a more advanced optimization phase that employs the tabu search metaheuristic. Tabu search⁵ is an advanced optimization technique that addresses the challenge of local optima by using a "tabu list" to avoid previously explored solutions that are considered less optimal. As the algorithm searches for better solutions, it makes small changes to the current solution, evaluating neighboring options. If a neighboring solution is found to be superior and not on the tabu list, it becomes the new current solution. Moves remain on the tabu list for a predefined duration, preventing immediate re-exploration of those states. Once the time expires, the algorithm may revisit these moves, allowing for greater exploration of the solution space while achieving a balance between exploration and exploitation. This adaptive mechanism enables tabu search to effectively escape local optima and discover higher-quality solutions in complex optimization problems.

This two-phase algorithm, effectively combines the strengths of heuristic and metaheuristic approaches to deliver high-quality and efficient solutions tailored to the specific needs of complex route optimization problems

Key Metrics and Early Simulated Testing of Route Optimizer and Resource Allocation Services

The model generates a set of metrics that help analyze each routing solution in detail. For our case, we focus on key metrics like the duration of each route, the total time across all vehicles, and the total distances covered, as these are relevant for evaluating and optimizing our specific logistics operations. Additionally, we track the load management throughout each journey, which allows us to see how effectively each vehicle handles its pickups and deliveries, ensuring that vehicle capacities and operational limits are used optimally. The optimization module that has been developed, has been tested through simulated datasets in the location where Greek pilot Part C is expected to take place. In the real-

⁵ Glover, F. (1990). Tabu search: A tutorial. *Interfaces*, 20(4), 74-94.

world pilot, the deployment is likely to involve one UGV and one UAV. However, to better illustrate the capabilities of the algorithm, we showcase a more generic scenario that incorporates multiple vehicles of various types for delivery operations. Specifically, the scenario tested includes five vehicles, of three separate types: two “UGV_Type_1”, one “UGV_Type_2” and two “UAV_Type_1”. To simulate a realistic scenario, we have generated a set of random delivery locations within the NKUA campus. One of the testing examples contains a total of 12 delivery points at various distances ranging from 100 meters to 1,5 kilometers apart, to represent a range of potential delivery challenges. In our simulation, we evaluated two distinct optimization approaches—minimizing total distance and minimizing total time. In most cases, the minimization of cost is a combination of the above, depending on the specifications of the vehicles involved in the process. For each vehicle, the time and distance matrices were generated using the modules that calculate travel times and distances (sub-section 4.2) for all possible combinations of origin-destination locations, while taking into account constraints such as no-go zones and geofencing.

Details about the vehicles' characteristics and shipment information are provided in Table 1 and Table 2, respectively.

	UGV type 1	UGV type 2	UAV type 1
Max Duration [min]	40	40	35
Operational Range [km]	3	3	3.5
Weight Capacity [weight units]	15	10	20
Volume Capacity [volume units]	15	10	20
Unload Time [min]	1	2	1
Number of Lockers	3	5	6

Table 1: Vehicle specifications for the simulated testing example

No. Shipment	Weight [weight units]	Volume [volume units]
1	1	1
2	1	2
3	2	4
4	4	8
5	2	2

6	4	3
7	5	3
8	6	4
9	2	2
10	3	2
11	2	1
12	2	2

Table 2: Shipment details for the simulated testing example

Time and distance matrices are generated from the exact delivery locations, considering factors such as vehicle speeds and geographical restrictions. These matrices are then used by the solver to perform the optimization. For simplicity and to avoid unnecessary numerical details, the specific delivery locations along with the corresponding time and distance matrices for this example, are not provided in this document.

The optimization results are presented in Table 3 and Table 4 for the minimization of total distance, and in Table 5 and Table 6 for the minimization of total time.

Optimization goal	Minimize Total Distance
Total time for all routes	85 min
Total distance for all routes	9.493 km

Table 3: Overall summary for total distance minimization

	Shipments Sequence	Total Time [min]	Total Distance [km]	Total Weight [weight units]	Total Volume [volume units]
ugv_type_1 – No 1	-	-	-	-	-
ugv_type_1 – No 2	4, 3, 11	7	2.478	8	13
ugv_type_2 – No 1	2, 1	10	2.95	2	3
uav_type_1 – No 1	8, 7, 6	33	2.341	15	10
uav_type_1 – No 2	5, 9, 12, 10	35	1.724	9	8

Table 4: Detailed route results for total distance minimization

Optimization goal	Minimize Total Time
Total time for all routes	61 min
Total distance for all routes	10.567 km

Table 5: Overall summary for total time minimization

	Shipments Sequence	Total Time [min]	Total Distance [km]	Total Weight [weight units]	Total Volume [volume units]
ugv_type_1 – No 1	2, 1	8	2.95	2	3
ugv_type_1 – No 2	11, 3, 4	7	2.478	8	13
ugv_type_2 – No 1	9, 5, 12, 10	13	2.787	9	8
uav_type_1 – No 1	-	-	-	-	-
uav_type_1 – No 2	7, 8, 6	33	2.352	15	10

Table 6: Detailed route results for total time minimization

The optimization results clearly illustrate the impact of different objectives on the routing solution. When the goal is to **minimize total distance**, the total distance traveled across all routes is 9.493 km, and the total time is 85 minutes. Conversely, when the goal is to **minimize total time**, the total time for all routes decreases to 61 minutes, but the total distance increases to 10.567 km. This expected outcome highlights the trade-off between time and distance: optimizing for time reduces the overall duration of the routes but may lead to longer distances, while optimizing for distance minimizes the distance at the cost of increased time. Additionally, it’s clear that the results demonstrate how the solver respects the constraints for each route and vehicle. The optimization process effectively adjusts the route planning based on the chosen goal, ensuring that the capacities of each vehicle and the specific optimization objective—whether minimizing time or distance—are taken into account to produce a feasible and efficient routing solution. Finally, the results illustrate that not all available vehicles were utilized by the algorithm. This indicates that the solver can effectively optimize the number of vehicles needed for the given routes, potentially allowing for a reduction in fleet size while still meeting the constraints and optimization goals. A more detailed analysis and evaluation of the method, including specific performance metrics and comparisons between the different optimization approaches will be included in the second version of the deliverable, where data provided by the LC of the project are expected to be used for a more realistic evaluation. This

will provide further insights into the effectiveness of our optimization strategies and potential areas for improvement in future implementations.

4.1.3 Last Mile Logistics with Marsupial Systems

Optimization Problem Statement

In this scenario, we have an application of a heterogeneous autonomous system with cargo bikes which carry UAVs in areas where flights are not permitted and UAVs that make deliveries to inaccessible places by the cargo bikes (like the demonstration in Diagonale Verde of Modena). Such systems are referred to as marsupial in the literature [10, 11], offer better coverage of services in an area, and are utilized to retrieve UAVs. Moreover, the cargo bikes can form platoons to execute the deliveries efficiently, and they can split from a platoon and join another one, once they reach a point within a given set of intermediate points I . Respectively, the UAVs can take off and land in a set of designated points U . The aim of the autonomous system is to deliver parcels from a single depot to their destination optimally according to key performance indices.

Each parcel $p \in P$ has: a weight $w_p \in R^+$, and a volume $v_p \in R^+$. Moreover, the bikes have a battery status $s^{(bat)} \in R$, a maximum range $d^{(max)}$ from origin that is recommended to not be exceeded, a unit cost of electricity per distance $c \in R$ and a charge cost $c^{(charge)} \in R$. With $\lambda \in [0,1]$ we denote a safety factor for the battery consumption. The optimization goal is to minimize the cost of the delivery.

The solution

The development of an algorithm that calculates the optimal solution includes two steps. In the first step a directed graph $G = (V, A)$, is computed as an abstraction of the area of delivery. The set of vertices consists of three subsets, the initial depot, i.e., the origin O , the set of destinations D , and a set of intermediate points $V = \{O\} \cup D \cup I \cup U$. Figure 4 illustrates an example. Every arc $a \in A$ of the graph is related to four values:

$$(d_a, t_a, c_a, m_a) \in R^+ \times N \times R^+ \times \{0,1\}$$

which correspond to the distance, estimated travel time, energy consumption and mode of service (1 if the access is provided by cargo bikes, 0 otherwise) of the arc $a \in A$.

The set of cargo bikes will be denoted with B and the set of parcels for delivery with P . Running Dijkstra's algorithm provides the shortest paths from O . In other words, it provides for every parcel $p \in P$, a route $r = a_1^{(r)}, \dots, a_k^{(r)}, a_i^{(r)} \in A, i \in \{1, \dots, k\}, k \in N$, the set of routes will be denoted by R . Every route r is related to four values:

$$(d'_r, t'_r, c'_r, m'_r) = \left(\sum_{a_i \in r} d_{a_i}, \sum_{a_i \in r} t_{a_i}, \sum_{a_i \in r} c_{a_i}, \prod_{a_i \in r} m_{a_i} \right)$$

which corresponds to the distance, estimated travel time, energy consumption and mode of service (1 if the use of a UAV is not needed, 0 otherwise) of the route $r \in R$.

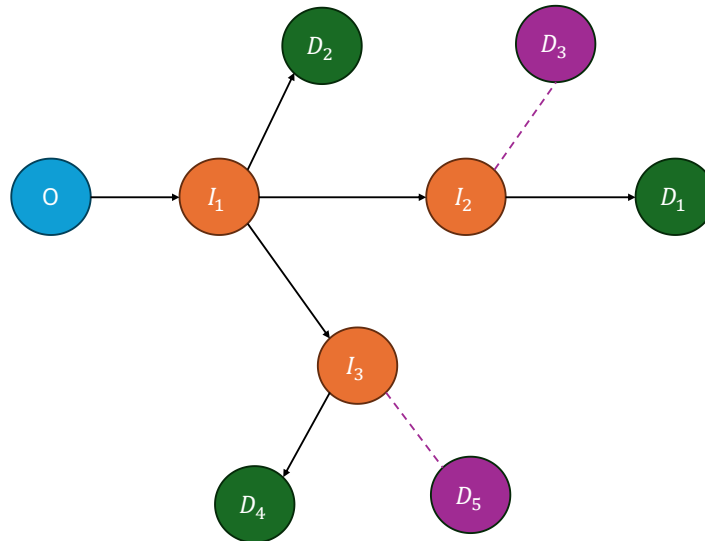


Figure 8:A graph representation of the intermediate and the delivery points

In this example, the graph consists of the starting point O , the set of intermediate vertices $I = \{I_1, I_2, I_3\}$ where the platooning of the cargo bikes may change, while $U = \{I_2, I_3\}$. The destinations D_1, D_2, D_4 which are reachable by bike, and D_3, D_5 which are reachable by a drone. In this case, we assume the graph is acyclic because the cargo bikes operate on dedicated bike roads. By assigning each vertex in the graph a unique string identifier, we can identify and combine routes that share multiple common edges—defined as matching substrings up to a specific threshold length.

For instance, referring to Figure 4, suppose we have two routes: “ $O, I_1, I_2, D_3, I_2, I_1, O$ ” and “ $O, I_1, I_2, D_1, I_2, I_1, O$ ”. Given that they share the common segment “ $O, I_1, I_2,$ ” we can merge these routes into a single combined route: “ $O, I_1, I_2, D_3, I_2, D_1, I_2, I_1, O$ ”.

Incorporating these combined routes allows us to define a matrix M_{pr} , where:

$$M_{pr} = \begin{cases} 1, & \text{if parcel } p \in P \text{ can be delivered via } r \in R \\ 0, & \text{otherwise} \end{cases}$$

This matrix M_{pr} enables the efficient mapping of parcels to optimal delivery routes, ensuring that shared path segments are utilized to their full potential, ultimately improving route efficiency and reducing redundancy.

The calculation of the routes concludes the first step.

Mathematical Model for the Mixed Integer Linear Program

In the second step, an integer linear program decides the optimal schedules for the bikes. The mathematical modeling of the above problem is as follows.

Variables:

$$x_{prb} = \begin{cases} 1, & \text{if parcel } p \in P \text{ travels via route } r \in R \text{ with the bike } b \in B \\ 0, & \text{otherwise} \end{cases}$$

$$x_{rb} = \begin{cases} 1, & \text{if bike } b \in B \text{ travels via route } r \in R \\ 0, & \text{otherwise} \end{cases}$$

$$y_r = \begin{cases} 1, & \text{if route } r \in R \text{ is selected} \\ 0, & \text{otherwise} \end{cases}$$

Subject to the following Constraints:

Route Parcels Consistency:

If the parcel cannot be delivered via a route, then the variable should be zero.

$$x_{prb} \leq M_{pr}, \quad \text{for every } (p, r, b) \in P \times R \times B$$

If a route is selected, then all the parcels that correspond to this route are delivered.

$$\sum_{b \in B} \sum_{p \in P} x_{prb} = \sum_{p \in P} M_{pr} y_r, \quad \text{for every route } r \in R$$

Send all parcels:

$$\sum_{r \in R} \sum_{b \in B} x_{prb} = 1, \quad \text{for every } p \in P$$

Bike and Route Consistency Constraints:

The following constraints ensure that if no shipments are assigned to a vehicle, then the corresponding variable is set to 0.

$$x_{rb} \leq \sum_{p \in P} x_{prb}, \quad \text{for every } (r, b) \in R \times B$$

$$y_r \leq \sum_{b \in B} \sum_{p \in P} x_{prb}, \quad \text{for every } r \in R$$

The following constraints ensure that when at least one shipment is assigned to a vehicle then the variable that corresponds to its selection is set to 1.

$$x_{rb} \geq x_{prb}, \quad \text{for every } (p, r, b) \in P \times R \times B$$

$$y_r \geq x_{prb}, \quad \text{for every } (p, r, b) \in P \times R \times B$$

Capacity Constraints:

$$\sum_{r \in R} \sum_{p \in P} x_{prb} \leq N_b, \quad \sum_{r \in R} \sum_{p \in P} x_{prb} w_p \leq W_b, \quad \sum_{r \in R} \sum_{p \in P} x_{prb} v_p \leq V_b,$$

for every $b \in B$

The Maximum Allowed Range should not be exceeded:

$$\sum_{r \in R} x_{rb} d_r \leq d_b^{(max)}, \quad \text{for every } b \in B$$

Energy Consumption Constraints:

$$\sum_{r \in R} x_{rb} c_r \leq \lambda s_b^{(bat)}, \quad \text{for every } b \in B$$

Objective Function: The objective function minimizes the cost of the total consumed energy of the bikes

$$\left(\sum_{b \in B} c^{(charge)} \sum_{r \in R} x_{rb} c'_r \right)$$

Using the variable x_{prb} , the algorithm returns an optimal schedule and routes for the delivery of the parcels. For the implementation of this algorithm Gurobi Optimizer was utilized.

4.2 Optimal Path Finding Algorithms

As mentioned in the previous section, finding optimal schedules and routes entails knowing the cost of traveling from one address to another. This section introduces two algorithms, for aerial and ground vehicles, to calculate the optimal path from point A to point B, alongside its distance and estimated travel time. The first developed algorithm calculates the shortest paths for ground vehicles by avoiding no-go zones (excluded area of the road network), while the second calculates flight paths for aerial vehicles within an operational area that contains no-fly zones.

4.2.1 Ground Vehicles

An algorithm has been designed to find an optimal path that inputs a list of addresses (or coordinate points) and provides the pairwise shortest paths (sets of waypoints) alongside their travel distance and estimated travel time. The algorithm's output is the input of the algorithms introduced in Section 4.1.

OpenStreetMap⁶ (OSM) is selected to obtain information about the road network for various modes of transportation within the operational area. As this algorithm is intended to guide multiple types of vehicles, manned or autonomous, it includes as part of the input no-go zones, areas within the operational area that vehicles must avoid.

Input: In particular, the input of the algorithm is:

1. **A set of destinations:** addresses that the algorithm will route between.
2. **List of exclusion polygons:** a list of polygonal areas representing geographic regions that must be avoided in routing. Each polygon is defined by a set of geographic points (latitude and longitude pairs) that form the polygon's boundary.

⁶ <https://www.openstreetmap.org/about>

3. **Transport Mode:** This parameter specifies the desired mode of transportation for the routes and controls the type of road network used in the graph. Acceptable values include:

- "drive" for car routes,
- "walk" for pedestrian routes,
- "bike" for cycling routes, and
- "all" for all road types, including paths inaccessible by car.

Output: The algorithm's output contains information on the pairwise routes between the specified destinations. Each pair of addresses corresponds to a set of waypoints (the path), the travel distance, and the estimated time travel.

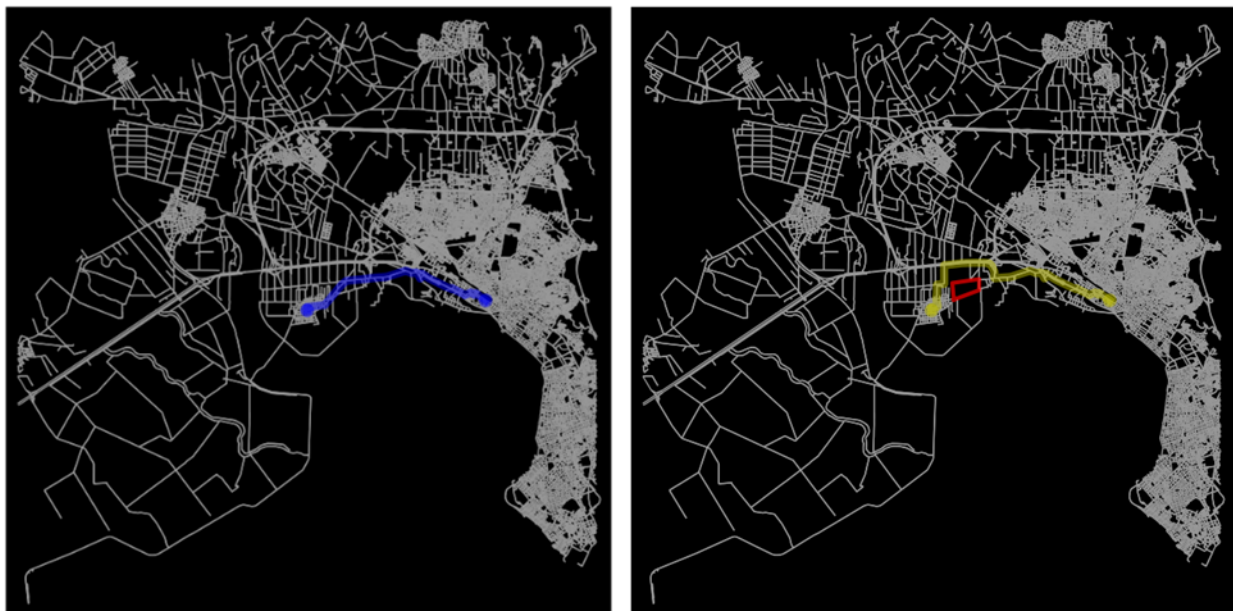


Figure 9: An example of a change in the shortest path after excluding the polygonal area.

The Algorithm's High-Level Overview

The algorithm first translates destination addresses into geographic coordinates through geocoding and then uses OSM data to generate a navigable graph structure based on the specified transportation mode (e.g., driving, walking, biking). The algorithm begins by extracting a region-specific OSM graph centered around the geometric center of the destinations and deletes all the edges (road

segments) within the exclusion zones, defined as convex polygons representing restricted areas on the map.

For each unique pair of destinations, the algorithm determines the nearest OSM nodes to the coordinates and applies Dijkstra's shortest path algorithm⁷ on the graph to find the minimum-distance route (see Figure 5). This route is characterized by its sequence of waypoints, total travel distance, and estimated travel time. It is computed by assigning speeds to road types, either based on OSM's maximum speeds (if available) or default speed values for various road classes.

4.2.2 Aerial Vehicles

While several open source or commercial solutions exist for the generation of the most efficient route between two points, providing distance, estimated time, and navigational waypoints, for cars, pedestrians, or cyclists, it is not common to find similar solutions for use with UAVs. However, the use of such component in the context of T4.4 is necessary, to generate the distance and time matrices that will be used in the route optimization procedures, and to provide navigational instructions for the unmanned aerial vehicles (UAVs) participating in the logistics operations to follow. This way, in the context of this task was generated from scratch a sub-module that provides this functionality.

The major requirement for the development of this component was to support geo-fenced operations. To do that, it was decided that the user should be able to define both a geo-fenced operational zone, where the UAV will be able to move freely inside it, and additionally, no-fly-zones inside this region. An indicative example of trajectories following this approach is shown in Figure 10.

⁷ Cormen, Thomas H., Leiserson, Charles E., Rivest, Ronald L., Stein, Clifford **"Introduction to algorithms"**, 2022 , MIT press

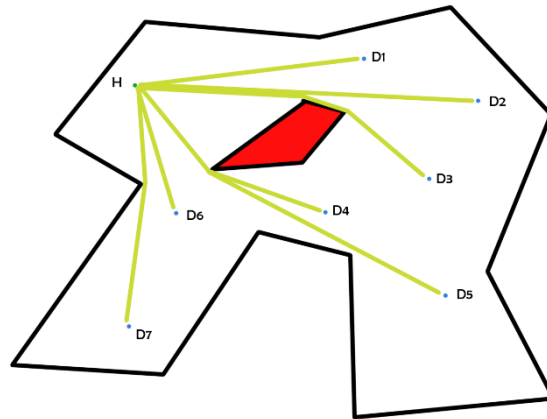


Figure 10: Geo-fenced zone with no-go-zones inside for the route generation of the UAVs

Additionally, based on CERTH’s long experience with UAV operations, it was decided that for safety reasons, the UAV routes should follow the approach of taking off vertically, reaching a user-defined transitional altitude (where it is considered that the operation can be obstacle-free, and thus safe from collisions), keeping this altitude until they get right above of the destination position, and landing vertically to this position (to perform the programmed parcel’s delivery or pick-up), as shown in Figure 11.

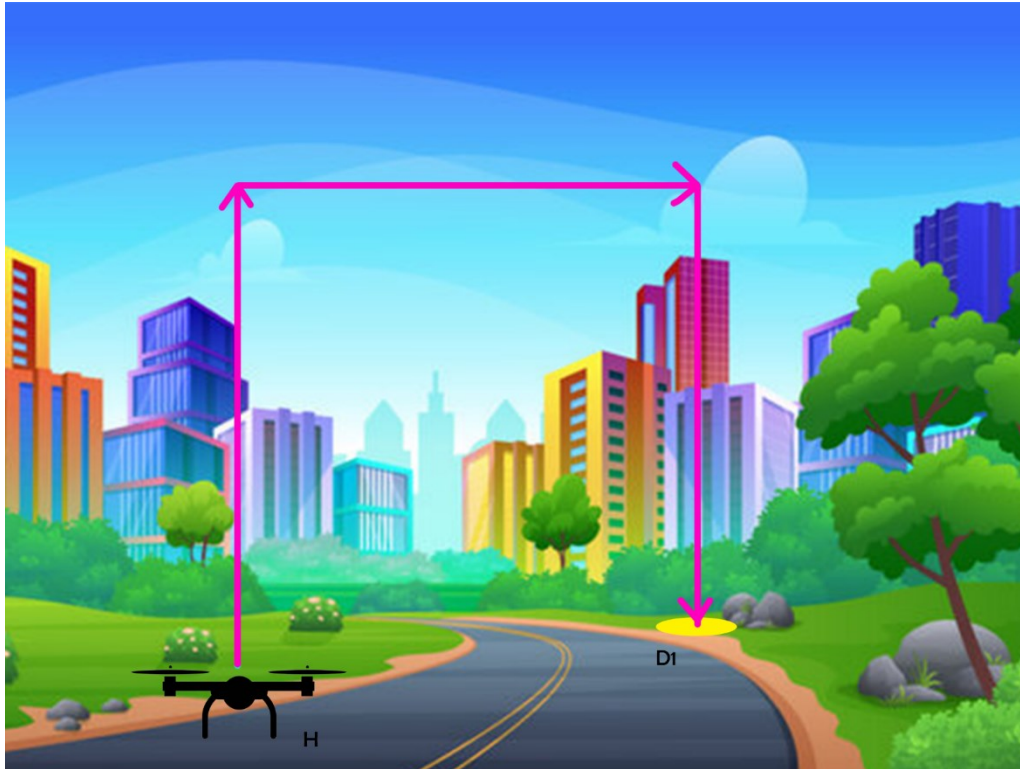


Figure 11: Vertical take-off and landing for UAVs, with a transition among positions in a user-defined constant altitude

Overall, the UAVs’ route generation component that was built, receives as input:

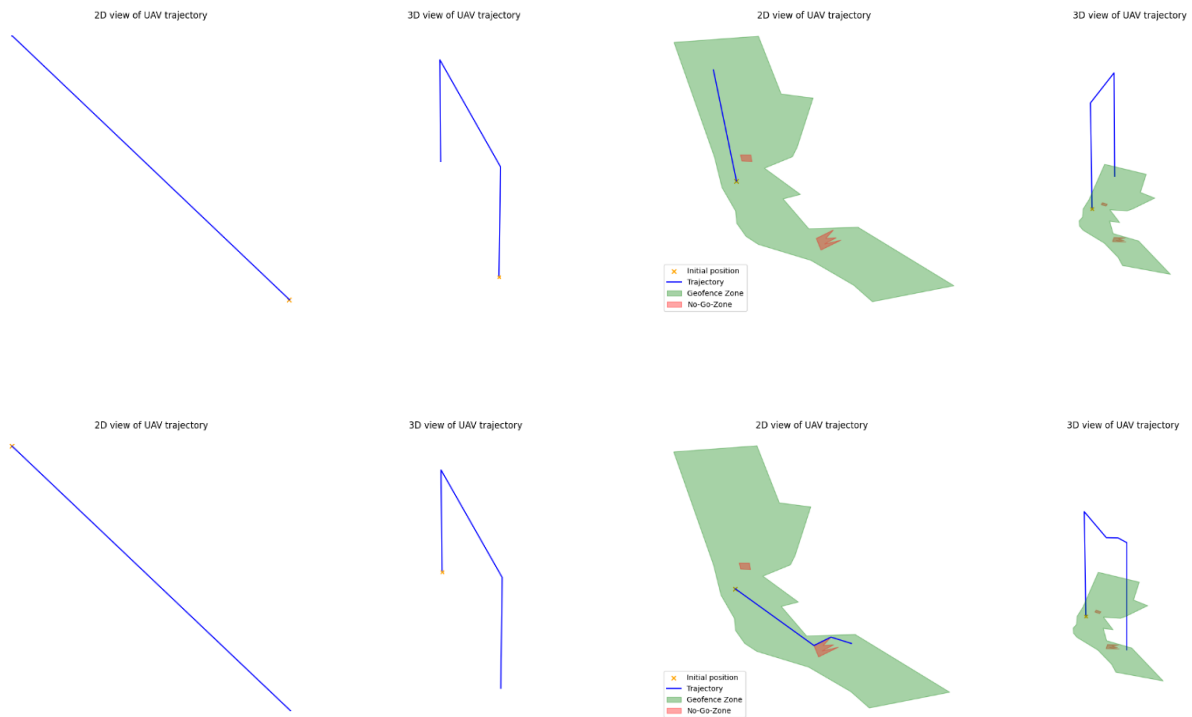
- An optional **geo-fenced zone**, with the presence of also optional **no-fly zones** inside it, formatted in the WGS84 coordinate system.
- A set of (at least two) **locations** that the UAV should visit, formatted in the WGS84 coordinate system as well.
- The designated **transitional altitude** among waypoints to visit.
- The desired **speed**, as well as the **maximum vertical speed** (an average of maximum ascending and descending speeds), a parameter needed for the time estimation procedure of the UAV flights.
- A **use cost per hour**, for the cost estimation of the UAVs’ operation.

Having received this information, it generates and provides as output:

- The **trajectory** among given positions, as a series of WGS84 coordinates, with the provided order.
- The trajectories’ overall **distance**.
- The estimated flight’s duration (**time**).

- The operation's **cost**.
- **Distance and time matrices** to be used by the routing optimization modules.
- **2D and 3D visualization** of the trajectories.

Figure 10 presents some examples of trajectories generated by the UAV route generation component. The left column shows the trajectories generated for some positions (in 2D and 3D view) considering that no geo-fencing is enforced, while the right column shows the trajectories among the very same positions, this time given a geo-fencing zone with no-fly zones inside it. The first two examples present trajectories among two positions, while the last one presents the complete trajectory to visit a series of designated positions. It is worth mentioning that non-convex polygons are supported (as shown in the example) both for allowed and forbidden zones, and additionally, that the area presented in this example corresponds to the location that the Greek Pilot Part C is expected to take place. Similarly, the route generation service can provide trajectories and support geo-fenced operations in any user-defined location, supporting both convex and concave polygons for the definition of permitted and no-go-zones.



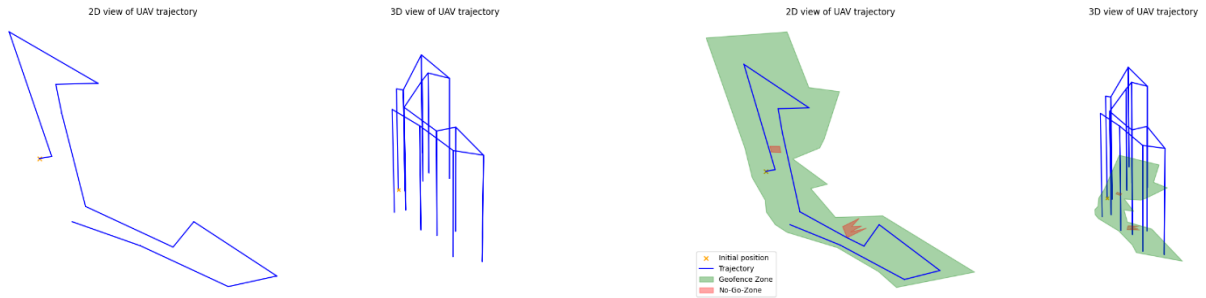


Figure 12: Trajectory examples in free space and geo-fenced zones

Trajectory Generation

To generate a trajectory between two given positions in a non-geo-fenced area, the concept is very simple. Supposing that the positions are [lat1, long1], [lat2, long2], the generated trajectory will contain the following waypoints:

[lat1, long1, 0]

[lat1, long1, transition_altitude]

[lat2, long2, transition_altitude]

[lat2, long2, 0]

Given the presence of geo-fencing zones and no-go zones, among the second and third waypoints of the sequence should be included additional waypoints (with the designated transition_altitude), inside the allowed area, such that the trajectory does not intersect any of the user-defined polygons. To achieve this, a combination of checks about if the generated lines intersect any of the polygons, and use of algorithms for the efficient generation of the shortest path among given positions is performed. In the provided implementation, the additional nodes (positions) that can be introduced to the non-geo-fenced trajectory can be placed on any of the sides of the polygons. It is critical that when geo-fencing is desired, all designated positions to be visited are always placed inside the allowed areas.

Distance, Time, and Cost Calculations

Having the exact trajectory to be followed, the overall distance is calculated as a sum of the distances for all line segments that constitute the trajectory. Regarding the time estimation, the concept is that the linear time for this distance is calculated, and a delay per waypoint is added. For this calculation,

a weighted average of the designated speed and the maximum vertical UAV speed (if the designated one exceeds this value) is used. Additionally, the delay that is added in each waypoint is parametrical to the UAVs’ speed and limited by a configurable sigmoid function. Moreover, for each destination position, an additional delay, corresponding to the time that will be needed for the loading and unloading of parcels is accounted. The time estimation procedure is configurable with variables, and the parameters for a more precise time estimation will be performed after the field trials. Finally, given the estimated time of a flight is calculated the operation’s cost, a parameter that solely depends on the time that each UAV is used.

4.3 Routing and Scheduling Services in the TRACE System

Figure 13 shows the latest version of TRACE Architecture (at the moment this deliverable is written). The “Intelligent Services for Logistics Optimization”, as described in this section, are placed in the blue bounding box, and are developed in the context of “T4.4 - Logistics Operations and Intelligent Scheduling”. This architectural figure is included here for the better understanding of the following two sub-sections, regarding the integration and compatibility of the work described in this section with the existing logistics frameworks, and its compliance and alignment with the project’s architecture and the defined requirements.

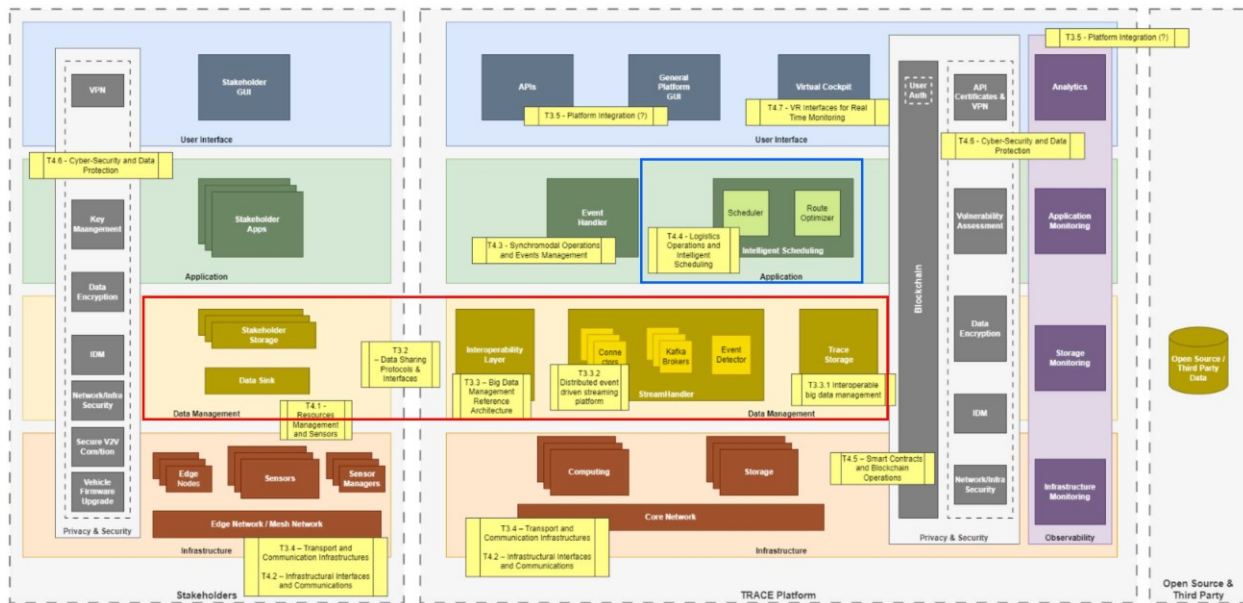


Figure 13: T4.4 in TRACE Architecture

4.3.1 Integration with Existing Logistics Frameworks

The red bounding box of Figure 13 includes the components and modules of TRACE architecture that are responsible for receiving data from the logistic companies, transforming them to a format compatible with the TRACE project's data model, and combining all the appropriate information from different sources to share them with them within the appropriate TRACE modules and services. The work performed by the components contained in this box ensures the compatibility and robust integration of the work performed within T4.4, with the existing frameworks and infrastructure of logistic companies. All implementations within T4.4 have been performed with a modular design that supports the out of the box integration of both legacy and state-of-the-art solutions and resources of the logistic companies, while the internal architecture of T4.4 (as presented in the following sub-section) allows for relatively easy functionality expansions in the future to support different kind of operations, means and infrastructure. This way, any existing logistics frameworks can be integrated with TRACE project, by building the appropriate data transformer and connector (T3.2), allowing all companies to exploit the functionalities and efficiency benefits of this ambitious system, while the modular design and build of the system ensures its ability to get extended to support additional functionalities, making it futureproof and compatible with upcoming demands and standards.

4.3.2 Alignment with Project Architecture and Requirements

Two platform components are responsible for addressing the optimal scheduling and routing. One is the **Scheduler**, and the other is the **Route Optimizer**. The Route Optimizer is responsible for calculating the shortest paths with their estimated travel time, and distance. In particular, the Scheduler is central in coordinating shipment requests and optimizing delivery operations. Upon receiving a shipment request from the User Interface, it retrieves the necessary data from the Interoperability Layer to compile the input required for optimal scheduling and routing. The Scheduler then forwards the relevant addresses to the Route Optimizer, which calculates the paths, travel distances, and estimated travel times. Using this data, the Scheduler computes optimal routes and schedules, storing them as a proposed schedule in both the Blockchain and Trace Storage. The proposed schedule is sent back to the User Interface for approval. Once approved, the User Interface notifies the Scheduler to finalize the schedule. The Scheduler initiates the creation of a smart contract in the Blockchain, ensuring transparency and accountability. Upon receiving acknowledgment from the Blockchain, the Scheduler stores the finalized schedule in TRACE Storage. After confirming the schedule's successful storage, the Scheduler returns the completed schedule to the User Interface, completing the process (see Figure 14).

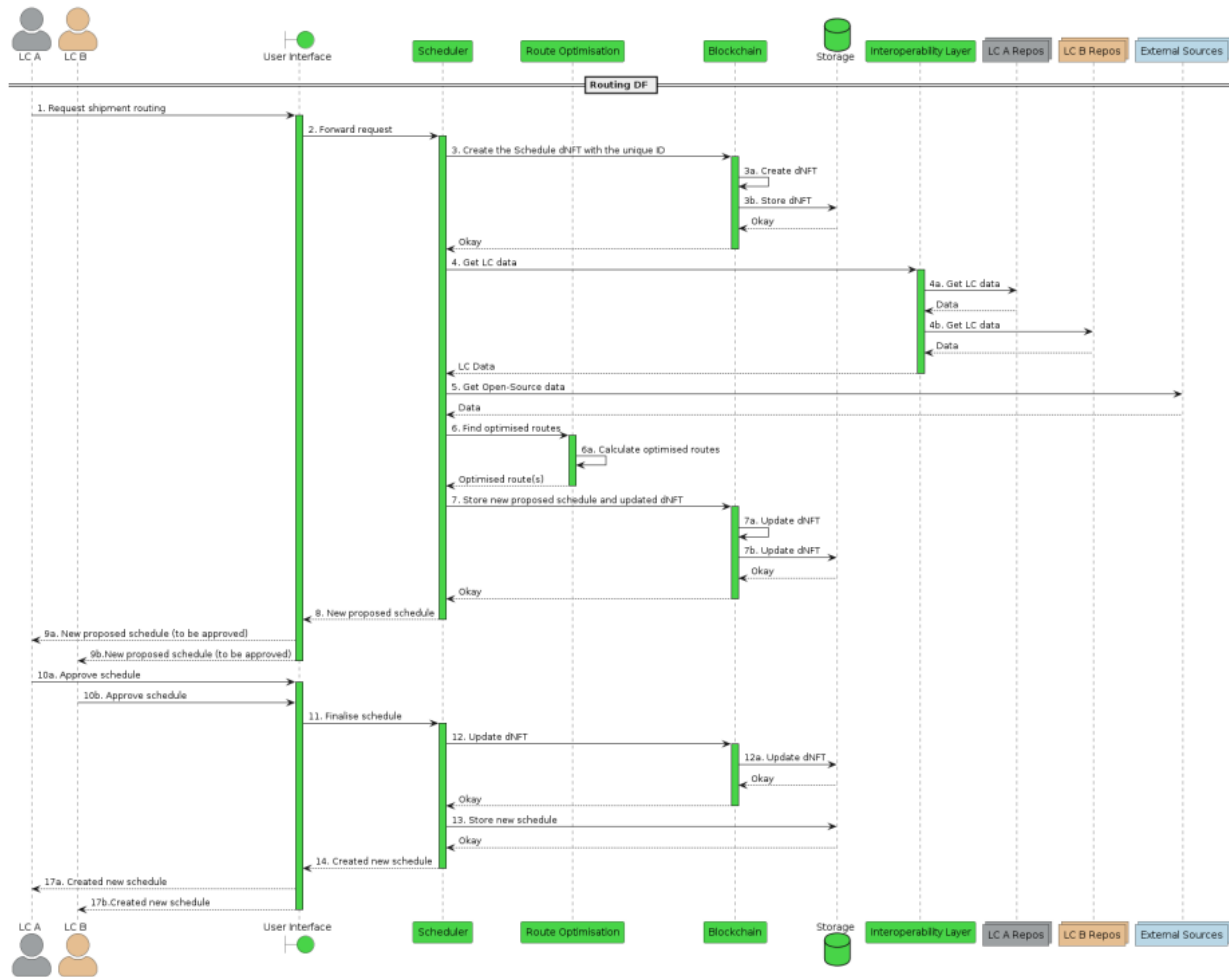


Figure 14: The flow diagram of the Scheduler’s and Route Optimizer’s functions during a request.

The Scheduler consists of four modules, as Figure 10 shows, to carry out all these functions. The modules are:

1. **The Wrapper:** As the name suggests, the wrapper wraps up all the functions of the Scheduler. It utilizes REST APIs to connect with the rest of TRACE’s components to receive or provide data and calls the appropriate solver according to the incoming request.

The connection with key TRACE components is established through the Wrapper’s designated integration points based on the Deliverable 3.1:

- **IP14_Sch-REManager:** When a delivery is disrupted, REManager uses the Wrapper to communicate with the Scheduler to dynamically send optimal routes to vehicles.
- **IP15_Sch-Blockchain:** Wrapper enables interaction between the Scheduler and the blockchain.
- **IP02_IntLayer-Sch:** This integration point facilitates the connection between the Scheduler and the Interoperability Layer to provide the input data required to calculate optimal solutions.
- **IP03_SH-CDMS:** The Wrapper ensures the permanent storage of data flowing through Streamhandler topics in the TRACE storage system.

These integration points underscore the Wrapper's critical role as a bridge between the Scheduler and other TRACE components, enabling efficient data flow, seamless communication to meet dynamic logistic challenges.

2. The Middle Mile Optimizer contains the algorithms presented in Section 4.1.1.
3. The First Mile/ Last Mile Optimizer contains the algorithms presented in Section 4.1.2.
4. The Last Mile in a Marsupial System with Platooning contains the algorithms presented in Section 4.1.3.

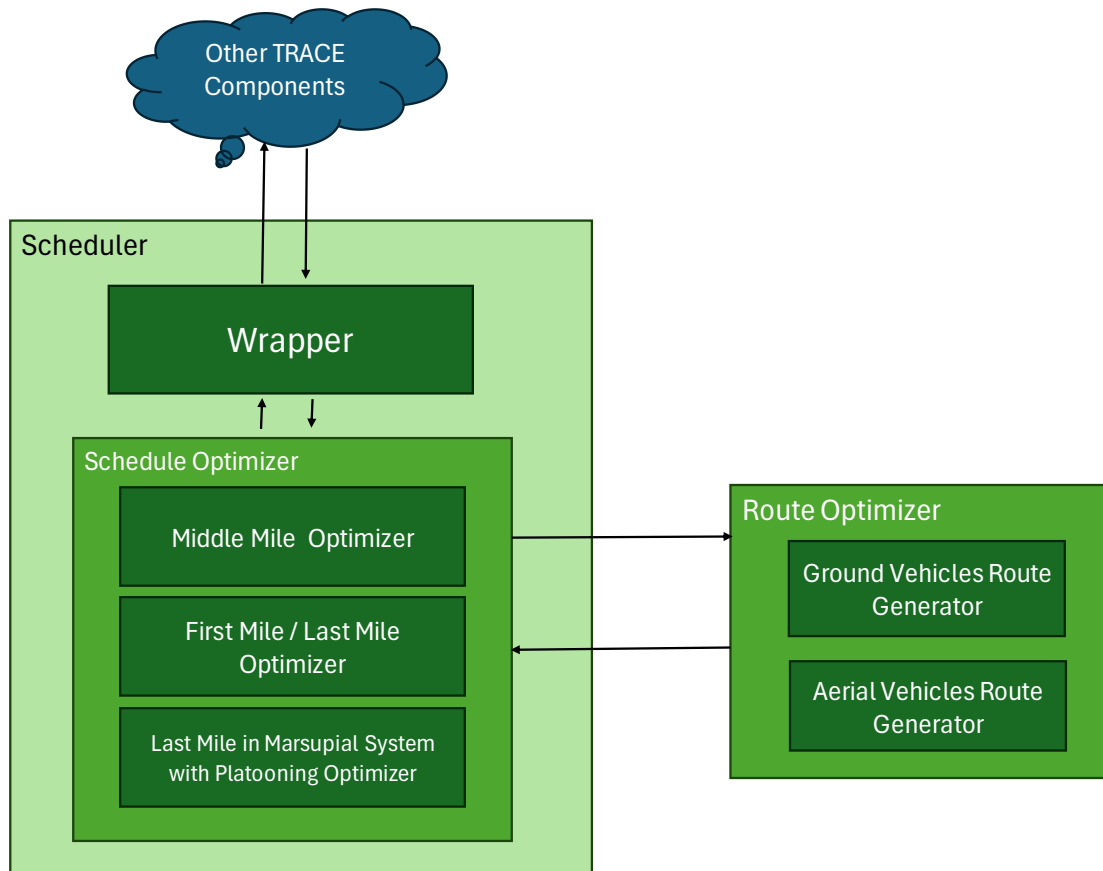


Figure 15: The Scheduler and Route Optimizer with their internal architecture.

Respectively, the Route Optimizer consist of two modules:

1. The Ground Vehicle Route Generator contains the algorithm of Section 4.2.1.
2. Aerial Vehicle Route Generator contains the algorithm of Section 4.2.2.

The design of two distinct modules is due to the inherently different requirements for path generation between ground and aerial vehicles. Ground vehicles navigate on predefined road networks and must account for factors such as road types and permissible routes. In contrast, aerial vehicles operate in three-dimensional space, and their flight path is affected by altitude, air traffic regulations, and safety protocols.

Additionally, no-fly zones for aerial vehicles and no-go zones for ground vehicles point to distinct algorithmic approaches. No-fly zones, often defined by regulations or safety concerns, require algorithms that calculate alternative flight paths while adhering to altitude constraints and restricted airspace boundaries. Similarly, no-go zones for ground vehicles, such as construction sites or areas with restricted

access, require specialized algorithms to reroute around these obstructions while maintaining efficiency and compliance with road network constraints. In conclusion, this design ensures that each vehicle type receives optimized, context-specific path planning tailored to its operational requirements.

The requirements that the Scheduler satisfies, as described in TRACE D2.1 are the following:

1. PLT-FUN-008 - TRACE platform shall dynamically adjust resource allocations, such as vehicle assignments or route schedules, based on real-time performance data and predictive insights to optimize efficiency and responsiveness
2. MON-PRM-002 - TRACE platform shall consider cost optimization objectives alongside performance metrics, balancing operational efficiency with cost-effectiveness to maximize overall value and profitability
3. MON-PRM-003 - TRACE platform should allow users to define performance thresholds and alerting rules to notify them when performance metrics exceed or fall below acceptable levels, triggering proactive intervention
4. RA-PRM-001 - TRACE should provide comprehensive performance tracking
5. RA-PRM-002 - TRACE may allow benchmarking and performance comparison

The requirements that the Route Optimizer satisfies are the following:

1. PLT-FUN-006 - TRACE platform should enable the planning of necessary measures to prevent or mitigate the negative impacts of future events
2. MON-FUN-010 - When generating trajectories and paths for unmanned vehicles, TRACE shall generate trajectories and paths crossing exclusively the authorized operating areas

5 Blockchain Infrastructure for Smart Contracts

5.1 Overview of Blockchain Technology

Blockchain is a decentralized digital ledger technology that provides unparalleled security, transparency, and immutability for transaction records. It has revolutionized industries by eliminating intermediaries and fostering trust among stakeholders through decentralized and tamper-resistant systems. Blockchain technology offers a groundbreaking approach to modernizing logistics by enhancing transparency, security, and efficiency. Unlike traditional systems, which often rely on centralized databases prone to errors and fraud, blockchain provides a decentralized ledger that records every transaction in a secure and immutable way. This ensures that all parties in the supply chain have access to a single, verified version of the truth in real time. The novelty of blockchain in logistics lies in its ability to enable secure, automated, and traceable transactions between various stakeholders such as suppliers, manufacturers, carriers, and consumers. Smart contracts—self-executing contracts with the terms directly written into code—can automate processes, reducing the need for intermediaries and speeding up transactions. Additionally, blockchain's immutability ensures that data cannot be tampered with, significantly reducing the risk of fraud or errors. With decentralized storage, blockchain can also enable secure and efficient management of data related to inventory, shipping, and delivery without relying on centralized authorities. This is particularly beneficial for cross-border logistics, where multiple jurisdictions and organizations must collaborate, and trust is often a major barrier.

In the TRACE project, blockchain plays a pivotal role in establishing trust between logistics partners by maintaining an immutable record of shipment data, key events, and transactions. This decentralized system eliminates the need for a central authority, significantly reducing the risks of fraud, tampering, or data loss, while enhancing the overall reliability of supply chain operations. Among the available blockchain platforms, TRACE adopts Algorand⁸ due to its distinct advantages in terms of performance, cost, and sustainability. While Ethereum offers robust smart contract capabilities, its high gas fees and scalability limitations during peak demand make it less suited for real-time logistics. Solana, though known for its high throughput, has faced concerns over its degree of decentralization and occasional network outages. Similarly, Cardano provides a strong focus on formal verification for security

⁸ <https://algorand.co/algokit>

but has a slower pace of updates and adoption. In contrast, Algorand delivers a balanced and optimized solution tailored for the needs of large-scale logistics.

Algorand's Pure Proof-of-Stake (PPoS) consensus mechanism ensures high throughput, processing up to 1,000 TPS with finality achieved in under five seconds, outperforming Ethereum (~15 TPS), Avalanche (~5 TPS), and Polygon (~38 TPS) in transaction scalability. This rapid processing is critical for logistics operations, where real-time updates are essential for maintaining efficiency. Moreover, its ultra-low transaction fees, typically around \$0.001 per transaction, ensure cost efficiency, enabling scalability without the financial overhead observed in Ethereum-based solutions. A standout feature of Algorand is its environmental sustainability. Unlike Bitcoin's energy-intensive Proof-of-Work (PoW) mechanism or Solana's hybrid approaches, Algorand's PPoS consumes minimal energy, making it one of the most eco-friendly blockchain platforms available. This is especially crucial in a time when businesses are under increasing pressure to adopt greener practices. By leveraging Algorand, TRACE assigns a unique dynamic Non-Fungible Token (dNFT) to each shipment, capturing its journey through the supply chain. A dNFT is a type of NFT that can change its properties or metadata over time based on external factors, such as user interactions, real-world events, or programmed conditions. Unlike static NFTs, which remain fixed, dynamic NFTs evolve, making them more interactive and adaptable for use cases like gaming, digital art, and real-time data integration. These dNFTs are updated in real-time via smart contracts, which automatically execute predefined conditions, such as recording a shipment's status at various checkpoints. This innovation enhances traceability, enabling stakeholders to monitor shipments securely and transparently. Furthermore, it strengthens accountability by creating a verifiable audit trail, ensuring that discrepancies can be quickly identified and resolved. With Algorand's advanced capabilities, TRACE not only optimizes supply chain management but also sets a new standard for transparency, efficiency, and environmental responsibility in the logistics sector. For more information, explore resources on [CoinTelegraph⁹](https://cointelegraph.com/learn/articles/what-is-algorand) and [Algorand vs Solana vs Venom: An In-Depth Blockchain Comparison¹⁰](https://www.techworm.net/2023/04/algorand-vs-solana-vs-venom.html)

⁹ <https://cointelegraph.com/learn/articles/what-is-algorand>

¹⁰ <https://www.techworm.net/2023/04/algorand-vs-solana-vs-venom.html>

5.1.1 Technical Overview of Algorand

Algorand is a cryptocurrency and blockchain protocol designed to overcome the scalability, security, and decentralization challenges faced by earlier systems such as Bitcoin. By eliminating forks, minimizing transaction latency, and achieving high throughput, Algorand offers a robust solution suitable for large-scale decentralized applications. The following sections provide a detailed breakdown of its architecture and core innovations.

5.1.1.1 Byzantine Agreement Protocol (BA★)

The cornerstone of Algorand's design is its Byzantine Agreement protocol, BA★, which allows participants to agree on the next block of transactions in a fault-tolerant manner. Unlike traditional protocols, BA★ dynamically selects participants, ensuring that no fixed set of users can be consistently targeted. It operates in two key phases:

1. **Reduction Phase:** Reduces the set of proposed blocks to two candidates, narrowing down options efficiently.
2. **Binary Agreement Phase:** Reaches consensus on a single block or an empty block using a sequence of voting steps.

This protocol ensures safety by preventing conflicting forks and guarantees liveness even in scenarios with temporary network partitioning. BA★ achieves high performance by completing the consensus process in as few as four steps under strong network synchrony.

5.1.1.2 Verifiable Random Functions (VRFs)

Scalability in Algorand is made possible through Verifiable Random Functions (VRFs), which enable the private and cryptographically secure selection of participants for consensus. Each user can determine their selection status by independently computing a function of their private key and the blockchain state. VRFs offer two critical advantages:

- **Randomized Selection:** Ensures fairness and unpredictability, preventing adversaries from preemptively targeting participants.

- **Proof of Participation:** Selected users provide cryptographic proof that their roles were legitimately assigned. By using VRFs, Algorand scales consensus efficiently to millions of users without introducing significant computational or communication overhead.

5.1.1.3 Weighted Voting and Sybil Resistance

To ensure robustness against Sybil attacks, Algorand employs a weighted voting mechanism, where a participant's influence is proportional to their stake in the system. This design guarantees that as long as honest users control more than two-thirds of the stake, the system remains secure and resistant to adversarial manipulation. Unlike traditional Byzantine Agreement protocols, which rely on a fixed set of nodes, Algorand's weight-based voting adapts dynamically to user participation.

5.1.1.4 Committee-Based Consensus

Consensus in Algorand is achieved through the use of dynamically selected committees. For each consensus step, a small subset of users is chosen to validate blocks and participate in voting. This committee-based design offers several benefits:

- **Efficiency:** Reduces the computational and communication load required for consensus by involving only a small fraction of participants in each step.
- **Security:** Prevents adversaries from targeting specific participants since committee membership changes unpredictably after every step.
- **Scalability:** Allows the system to handle large numbers of users while maintaining low latency.

Each committee member's selection is verifiable by other participants, ensuring transparency and trustworthiness in the consensus process.

5.1.1.5 Fork-Free Blockchain

Algorand guarantees that all honest users agree on a single, consistent blockchain without forks. This is achieved by the BA* protocol, which ensures that once a block is finalized, no conflicting block can be created. The elimination of forks offers several key advantages:

- **Faster Transaction Finality:** Transactions are confirmed within seconds, as users do not need to wait for multiple confirmations, unlike in Bitcoin.

- **Simplified State Management:** A single chain simplifies the verification and synchronization processes for new participants.

5.1.1.6 Cryptographic Sortition

A unique feature of Algorand is its cryptographic sortition mechanism, which uses VRFs to determine roles such as block proposers and committee members. Each user calculates their eligibility independently based on their private key and the current blockchain state. Sortition ensures:

- **Fair Role Assignment:** Users are selected proportionally to their stake, preventing adversarial domination.
- **Dynamic Role Reassignment:** Committee memberships are updated dynamically in each step, enhancing security against targeted attacks.

The use of sortition allows Algorand to avoid the inefficiencies of traditional consensus protocols while maintaining robust security guarantees.

5.1.1.7 Efficiency and Scalability

Algorand achieves high efficiency and scalability through its innovative consensus design. Experimental evaluations demonstrate:

- **Transaction Throughput:** Algorand achieves 125 times the throughput of Bitcoin by avoiding computationally expensive PoW and reducing the communication overhead.
- **Latency:** Transactions are finalized within 22 seconds in most scenarios, with minimal delay even under high user loads.
- **Scalability:** The protocol supports hundreds of thousands of users while maintaining constant transaction latency.

5.1.1.8 Resilience to Adversaries

Algorand is designed to withstand a wide range of attacks¹¹, including¹²:

- **Sybil Attacks:** Mitigated through weighted voting based on users' stake, ensuring that adversaries cannot amplify their influence by creating pseudonyms.
- **Denial-of-Service (DoS) Attacks:** The dynamic selection of committee members prevents adversaries from reliably targeting consensus participants.
- **Network Partitioning:** Safety is maintained under weak synchrony assumptions, while liveness is restored when the network regains strong synchrony.

5.1.1.9 Safety and Liveness Guarantees

Algorand provides strong theoretical guarantees for both safety and liveness. Under its "weak synchrony" model, the protocol ensures that:

- **Safety:** No two honest users will ever accept conflicting blocks, even under network asynchrony.
- **Liveness:** The blockchain continues to grow as long as the network is periodically synchronous.

These guarantees make Algorand a robust solution for applications requiring high reliability and fast transaction finality.

5.1.1.10 Bootstrapping and Recovery

Algorand simplifies the process of onboarding new users and recovering from network failures:

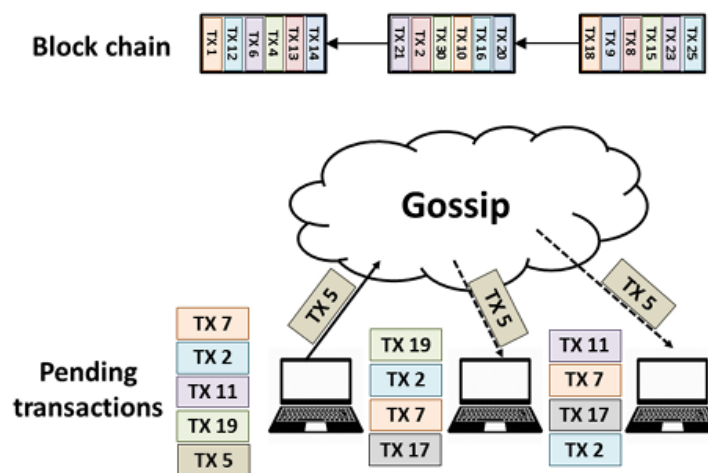
- **Bootstrapping:** New users can join the network by validating certificates associated with finalized blocks. These certificates aggregate the votes of committee members, allowing users to trust the blockchain state without downloading its entire history.
- **Fork Recovery:** In the event of a fork, Algorand employs a recovery protocol that uses loosely synchronized clocks and periodic checkpoints to ensure that all honest users converge on the same chain.

¹¹https://hackernoon.com/a-technical-qanda-on-network-partition-attacks-12283u5v?utm_source=chatgpt.com

¹² https://www.halborn.com/blog/post/exploring-the-security-of-algorand?utm_source=chatgpt.com

5.2 Smart Contracts for Real-Time Transactions

The TRACE platform utilizes a wide range of technologies to implement a secure and efficient system for tracking logistics data. At the core of its authentication mechanism is the use of Public Key Infrastructure for secure user login and identity management. Users generate a unique public-private key pair, which is essential for signing authentication requests without transmitting sensitive data over the network. The use of **Algorand’s SDK** (algosdk) ensures seamless integration of the blockchain with the TRACE platform. By leveraging **algosdk**, which is a JavaScript library designed for interacting with the Algorand blockchain. The system employs **nonce¹³-based authentication** to prevent replay attacks, enhancing security during the login process. The user’s private key is never exposed, and the nonce is used to generate a one-time signature, proving ownership of the wallet and ensuring the authenticity of each session. This method effectively eliminates the need for traditional passwords, reducing the surface area for hacking attempts. The **Axios¹⁴** library is used for API calls, facilitating smooth communication between components, ensuring real-time data exchange while interacting with external services like the blockchain.



An overview of transaction flow in Algorand.

Figure 16: An overview of transaction flow in Algorand

¹³ <https://bluegoatcyber.com/blog/nonce-understanding-the-concept-and-its-importance/>

¹⁴ <https://dev.to/ohdylan/a-simple-demonstration-of-api-call-using-axios-18fm>

TRACE can efficiently create and manage dynamic NFTs (dNFTs) associated with shipments. These dNFTs act as immutable records for each shipment, storing crucial metadata such as the shipment's weight, type, and tracking details. To store and manage immutable data, TRACE uses InterPlanetary File System **IPFS** through **Pinata's¹⁵ service** for decentralized file storage. IPFS is a decentralized, peer-to-peer file storage and sharing protocol designed to make the web faster, more secure, and more resilient. IPFS allows users to store and share files across a distributed network by addressing content based on its hash (content-based addressing) rather than its location (e.g., URLs). This approach ensures that data is immutable, tamper-proof, and accessible even if the original server is unavailable, making it a powerful tool for blockchain applications, decentralized apps (dApps), and file storage solutions. Ensures that shipment details and updates are stored in a tamper-proof, permanent manner. The metadata associated with each shipment is uploaded to IPFS, with each file's hash securely stored on the blockchain, enabling easy verification and traceability. This decentralized storage architecture, combined with TRACE Database, provides a hybrid solution that balances reliability with the flexibility needed to handle the high volume of logistics data.

In TRACE, the execution of smart contracts is driven by specific triggers based on the actions and events related to shipments. These triggers are defined within the system's backend infrastructure, using **Algorand's smart contracts (ASC1)**. ASC1 contracts are lightweight, secure, and highly scalable, designed to automatically execute predefined actions when certain conditions are met, such as updates to shipment status or changes in ownership. For example, when a logistics partner updates a shipment's status on the TRACE platform, the backend evaluates the data and checks if the conditions outlined in the smart contract are satisfied. If the conditions are met, the contract is triggered. This action can lead to updates to the dNFT representing the shipment, notifications to relevant stakeholders, or storing new information on the blockchain. The TRACE backend is central to triggering these smart contracts. It monitors the system for relevant events, such as a shipment reaching a new location, a change in ownership, or a transaction completing. Once such events occur, the backend communicates with the blockchain to execute the smart contract. Logistics partners—such as shipping companies—play an essential role in this process by providing real-time updates about the shipment's progress, including location changes, transit status, or delivery confirmation. These updates are sent to the TRACE platform, which then triggers the relevant

¹⁵ <https://www.pinata.cloud/blog/ipfs-for-decentralized-storage/>

smart contract on the blockchain. By utilizing blockchain technology, these updates are immutable, transparent, and secure, ensuring the integrity of the shipment's journey.

The system is designed to reduce human intervention and automate the process of updating shipment statuses and communicating with stakeholders. The smart contract's role is to enforce predefined conditions automatically, ensuring that every stage of the shipment journey is recorded on the blockchain and publicly available for all parties involved. By integrating real-time data from logistics partners and leveraging blockchain technology, TRACE ensures that shipment details are dynamically updated, transparent, and secure. Additionally, **Swagger UI**¹⁶ integrated with **swagger-jsdoc** enables the automatic generation of TRACE's API documentation. This provides clear, accessible, and user-friendly documentation for developers, making it easier to interact with TRACE's platform endpoints. The benefits of the chosen technologies are extensive. **Algorand's PPOS** consensus mechanism ensures fast transaction finality, low fees, and high scalability, which is crucial for real-time shipment tracking. **Decentralized storage**, facilitated by **IPFS**, provides permanent, tamper-proof records of the shipment journey, ensuring verifiability. **Axios** and **Pinata** streamline communication with external services, enabling smooth integration with third-party platforms. Moreover, the use of **smart contracts** automates the execution of logistics process conditions, reducing the need for manual oversight. Together, these technologies help TRACE create a robust, secure, and scalable platform for automated logistics management.

The following algorithm's decision-making process for blockchain consensus is used in the TRACE system's Blockchain module to mint a dNFT, making it tamper-proof.

¹⁶ <https://swagger.io/tools/swagger-ui/>

```

step ← 1
r ← block_hash
empty_hash ← H(Empty(round, H(ctx.last_block)))

while step < MaxSteps do
  CommitteeVote(ctx, round, step, τstep, r)
  r ← CountVotes(ctx, round, step, Tstep, τstep, λstep)

  if r = timeout then
    r ← block_hash
  else if r = empty_hash then
    for s' ← step+1 to step+3 do
      CommitteeVote(ctx, round, s', τstep, r)
    if step = 1 then
      CommitteeVote(ctx, round, final, τfinal, r)
    return r
  step++

  CommitteeVote(ctx, round, step, τstep, r)
  r ← CountVotes(ctx, round, step, Tstep, τstep, λstep)

  if r = timeout then
    r ← empty_hash
  else if r = empty_hash then
    for s' ← step+1 to step+3 do
      CommitteeVote(ctx, round, s', τstep, r)
    return r
  step++

  CommitteeVote(ctx, round, step, τstep, r)
  r ← CountVotes(ctx, round, step, Tstep, τstep, λstep)

  if r = timeout then
    if CommonCoin(ctx, round, step, τstep) = 0 then
      r ← block_hash
    else
      r ← empty_hash
  step++

// No consensus after MaxSteps; assume network
// problem, and rely on §8.2 to recover liveness.
HangForever()

```

Figure 17: Decision-making process for blockchain consensus of the TRACE system's Blockchain module

BinaryBA* algorithm, which is used for achieving consensus in Algorand blockchain system.

1. **Start:** Initialization of variables like `block_hash` and `empty_hash`.
2. **Committee Voting:** The algorithm performs voting through the committee (`CommitteeVote`) and counts votes (`CountVotes`).

3. **Timeout or Empty Hash:** If there's a timeout, the system reverts to `block_hash`. If `r = empty_hash`, additional votes are performed.
4. **Final Vote:** If at `step = 1`, a final vote is cast.
5. **Continued Voting:** The process continues until either consensus is reached or `MaxSteps` is reached.
6. **Timeout Handling:** If timeout occurs again, `CommonCoin` is used to decide between `block_hash` and `empty_hash`.
7. **Failure Handling:** If no consensus is reached after `MaxSteps`, the algorithm enters a `HangForever` state, indicating a network problem.

5.3 Financial Operations and Authentication

In TRACE's next phase after commercialization, financial operations are designed to seamlessly integrate with the existing logistics systems. The goal is to automate and secure financial transactions within the supply chain by leveraging blockchain technology, specifically Algorand's smart contracts. The financial operations within TRACE will be powered by smart contracts, which will trigger payment and invoicing actions based on specific conditions related to shipment deliveries. These contracts ensure that financial transactions are executed automatically once predefined terms—such as successful delivery or route completion—are met. For example, once a logistics partner confirms delivery, the smart contract will execute and release payments to the relevant parties, such as the shipping company or warehouse, ensuring that financial obligations are met on time and without manual intervention. By integrating Algorand's blockchain, the platform ensures transparency and immutability in all financial operations. Each transaction, including payments, settlements, and invoice creations, is recorded on the blockchain. This provides all stakeholders involved, from logistics companies to merchants, with a verifiable and auditable history of financial exchanges. The use of Algorand's PPoS mechanism ensures that transactions are processed quickly, efficiently, and at a low cost—key attributes when handling the high frequency of financial operations in a global logistics network. Furthermore, decentralized storage solutions like IPFS will be used to store financial documents and records in a tamper-proof and permanent manner. This ensures that all receipts, invoices, and payment histories are securely stored, easily accessible, and resistant to alteration, providing additional security and trust for users and partners alike. This automated, transparent financial system reduces human error and administrative overhead, increasing the efficiency of financial operations in logistics. By tying payments directly to shipment milestones, TRACE ensures that funds are only disbursed when services are fully rendered, protecting both buyers and suppliers. In

summary, the next phase of TRACE will enhance financial operations by leveraging blockchain's capabilities for secure, automatic, and transparent payments. With the integration of smart contracts, decentralized storage, and blockchain's immutability, the platform aims to streamline financial transactions and improve trust across the logistics ecosystem.

5.4 System/Subsystem Development

The TRACE blockchain infrastructure, built on Algorand's eco-friendly consensus mechanism, represents a next-generation decentralized architecture optimized for high-performance logistics and digital asset management. Leveraging Algorand's advanced Layer-1 capabilities, including secure and immutable transactions, Algorand Standard Assets (ASAs), and atomic transfers, TRACE ensures efficient and reliable operations for mission-critical applications. Its modular architecture integrates robust subsystems for wallet management, user authentication, NFT lifecycle management, and data handling, while smart contracts developed with TEAL and PyTEAL automate workflows and enforce business logic, ensuring transparency, compliance, and reduced manual overhead. Combining decentralized storage for redundancy with cloud-based infrastructure for high availability, TRACE delivers real-time analytics and system resilience under peak demands. Cryptographic techniques like multi-signature accounts and rekeying enhance security, while blockchain-native features such as transaction metadata fields provide an auditable history for accountability. Built with sustainability in mind, TRACE aligns with Algorand's low-energy protocol, making it an eco-friendly, scalable, and high-performance platform for decentralized logistics and asset tracking.

A. Wallet Integration

The wallet integration subsystem is a critical component of the TRACE ecosystem, facilitating the secure and efficient management of digital assets such as cryptocurrencies and NFTs. It supports wallet creation processes that utilize mnemonic phrases to ensure recoverability, adhering to deterministic key generation practices consistent with Algorand's standards. This guarantees compatibility with Algorand's tools and ensures robust security in key management. The subsystem also includes support for multi-signature wallets, catering to advanced use cases like corporate accounts or custodial services, where shared control over wallet access is necessary for enhanced security. To protect user assets, private key encryption is a fundamental requirement. The subsystem must ensure that private keys remain securely stored within the user's environment, employing advanced cryptographic techniques such as AES

encryption¹⁷ or secure enclave technologies. It is essential that private keys never leave the client-side environment, aligning with decentralized security principles. Non-functional requirements prioritize user experience and system reliability. The subsystem must deliver a user-friendly interface, simplifying complex blockchain interactions such as transaction signing and asset management for users without technical expertise. It should achieve low latency in operations, including wallet creation, transaction signing, and on-chain interactions, to ensure a seamless user experience even under high load. Furthermore, robust security measures must protect against threats like replay attacks, phishing attempts, and key compromise, ensuring the integrity of the entire wallet ecosystem. The architecture must be scalable, accommodating an expanding user base while maintaining performance and security standards.

The **system architecture** of the wallet integration subsystem acts as the fundamental interface connecting end-users with the Algorand blockchain. This architecture is responsible for ensuring seamless interactions between users and their digital assets while maintaining a high level of security. At its core, wallets within this subsystem are tasked with managing asset custody, including cryptocurrency balances and NFTs, and enabling users to perform blockchain operations such as transfers, asset creation, and smart contract interactions. To achieve this, the architecture integrates Algorand's JavaScript SDK as its primary tool for blockchain communication. This SDK serves as the bridge for constructing transactions, querying blockchain states, and interacting with Algorand-specific features like atomic transfers and rekeying. The subsystem also tightly interacts with the TRACE platform's blockchain interaction module, ensuring the efficient execution of asset transfers and transaction validations. The architectural design emphasizes the principle of decentralized security, ensuring that private keys remain encrypted and confined to the client-side environment, eliminating risks associated with key exposure on servers. Additionally, the architecture ensures that user interactions with wallets, whether through browser-based interfaces or mobile applications, remain smooth and responsive, irrespective of the underlying blockchain complexities.

The **subsystem design** of wallet integration is modular, comprising three core components: key management, transaction signing, and blockchain interaction. The **key management** component handles the secure creation, storage, and restoration of wallets. Wallets are generated using deterministic algorithms based on mnemonic phrases, allowing users to recover their accounts reliably while adhering to Algorand's cryptographic standards. This component also includes secure storage mechanisms for public-private key pairs, employing methods like encrypted local storage or hardware security modules

¹⁷ <https://csrc.nist.gov/pubs/fips/197/final>

(HSMs) for safeguarding sensitive credentials. The subsystem supports advanced key management features, such as rekeying, enabling users to update their private keys without altering their wallet addresses, thus enhancing security without interrupting usability.

The **transaction signing** component is responsible for ensuring that all transactions are securely signed using the user's private key. This process is carried out entirely on the client side, leveraging the Algorand JavaScript SDK to generate digital signatures that authenticate transactions while preserving the confidentiality of private keys. Features like multi-signature approvals are supported, enabling collaborative control over high-value wallets or organizational funds. This design also incorporates protection against replay attacks by embedding nonce and timestamp data in transactions.

The **blockchain interaction** component manages the communication between the wallet and the Algorand blockchain. It handles tasks such as broadcasting signed transactions, retrieving account states, and querying blockchain data for asset balances or transaction histories. This component ensures that interactions are optimized for real-time processing, enabling immediate feedback on transaction statuses. Furthermore, it integrates with TRACE's overarching blockchain interaction module, allowing seamless interoperability between the wallet subsystem and other blockchain-based functionalities of the platform, such as smart contract execution and event logging.

The implementation follows coding standards for security (e.g., using cryptographic algorithms for key management). It will utilize existing libraries for wallet generation and interaction with the Algorand blockchain. Wallet integration must be connected to the system's user interface and blockchain layers. Testing ensures wallet transactions, key management, and blockchain communication work smoothly. The wallet module is deployed within the broader TRACE infrastructure, ensuring secure connections between wallets and the blockchain. Configurations for wallet management, key storage, and API calls are set up to ensure smooth operation in the cloud or on-premises. After deployment continuous monitoring for security vulnerabilities, wallet updates, and changes to Algorand's SDK is required to ensure ongoing compatibility and user security.

B. dNFT Generation

The dNFT generation in the TRACE ecosystem creates unique digital tokens representing assets such as artwork, certifications, or physical ownership. This process adheres to ASA's protocols, ensuring interoperability and alignment with blockchain standards. Unique token identifiers are generated using

cryptographic algorithms and metadata, guaranteeing no two NFTs share the same identity. Metadata associated with each NFT is structured, immutable, and searchable, enhancing usability and traceability. The subsystem emphasizes high performance, security, scalability, and privacy, handling large-scale minting demands while protecting against unauthorized access and safeguarding sensitive information through encryption or selective disclosure. The NFT generation module integrates tightly with TRACE's blockchain and wallet systems, leveraging Algorand's JavaScript SDK for efficient token creation and management. Unique identifiers, combined with user-provided metadata like descriptions or external references, define each NFT. Metadata is stored securely using decentralized solutions like IPFS or Algorand's note field, balancing efficiency with immutability. The subsystem ensures on-chain registration of all NFT data, supports role-based access for minting, and ties ownership to wallet addresses. Advanced features, such as dynamic metadata or programmable royalties, are implemented through smart contract logic.

The design is modular, with dedicated components for token creation, metadata handling, and blockchain integration. The token creation module ensures unique, ASA-compliant NFTs, with optional configurations like transfer restrictions or reserve accounts. Metadata is validated for schema compliance and securely stored, while the blockchain integration component manages minting, ownership transfers, and real-time validation of NFT existence. Testing ensures NFTs are securely generated, transferred, and updated, with regular updates for compatibility with Algorand network upgrades and evolving NFT standards. This comprehensive architecture ensures TRACE's NFT generation subsystem is secure, scalable, and efficient, supporting its asset management capabilities.

C. NFT Updation

NFT updating in the TRACE ecosystem involves making modifications to the metadata or properties associated with an existing NFT, such as changing ownership details, updating asset descriptions, or altering the status of the token (e.g., marking it as redeemed or inactive). The system must ensure that any updates adhere to strict validity rules, preserving the integrity and authenticity of the original asset. For instance, changes to an NFT must not overwrite its unique identity or cryptographic signature. Updates must also be traceable, maintaining a history of all changes to the asset's state or metadata, enabling transparency and accountability. This is especially critical for NFTs linked to real-world assets, where provenance and compliance may be regulated or audited.

A key requirement is that updates must be authorized, ensuring only rightful owners or designated administrators can initiate changes. This entails robust mechanisms for verifying permissions using cryptographic signatures or multi-signature approvals. Non-functional requirements emphasize performance, with updates processed quickly and efficiently to maintain a responsive user experience. Additionally, security is paramount, requiring safeguards against unauthorized modifications, tampering, or replay attacks. To ensure data consistency, the subsystem must handle concurrent updates gracefully, preventing conflicts or duplication. The NFT updating subsystem is built to interact seamlessly with both the blockchain and the TRACE platform's data management components. It operates as an intermediary layer between user inputs and on-chain data, leveraging smart contract capabilities to enforce rules and execute updates. Using Algorand's JavaScript SDK, the subsystem facilitates the execution of asset update transactions on the blockchain. These updates may include modifying on-chain metadata fields, adjusting configurable properties (e.g., freeze status or transfer permissions), or transferring ownership of the NFT.

At the architectural level, the system employs a combination of on-chain and off-chain mechanisms to handle updates. Metadata updates are processed either directly on-chain, using Algorand's asset configuration features, or off-chain, where updated data is stored in a decentralized system like IPFS. In the latter case, the blockchain stores a hash of the updated metadata to ensure integrity and immutability. For ownership updates, the subsystem interfaces with the wallet integration module to validate the current owner's identity and confirm the transfer transaction before executing it on-chain. The architecture includes integration with the data management subsystem to ensure consistency across the TRACE ecosystem. For example, when an NFT's status or metadata is updated, corresponding changes must be synchronized with TRACE's centralized or distributed data repositories to maintain data integrity across user interfaces and related services. Role-based access control is embedded within the subsystem, allowing only authorized users or entities to initiate updates. This is implemented using Algorand's smart contract logic or multi-signature schemes, depending on the specific use case. By leveraging these architectural principles, the NFT updating subsystem ensures that updates are secure, efficient, and consistent while maintaining alignment with TRACE's broader blockchain infrastructure.

The NFT updating subsystem is meticulously designed to ensure that all changes to NFTs are secure, authorized, and compliant with predefined rules. This subsystem is responsible for processing updates to the metadata, properties, or ownership of NFTs while maintaining their integrity and

provenance. Its architecture integrates multiple layers of validation and interaction with the Algorand blockchain to guarantee that any modifications are reflected accurately on-chain.

At the core of the design is a robust **authorization mechanism**. Every update request is validated against a set of predefined rules to ensure compliance. For instance, ownership changes can only be initiated by the current owner or an entity explicitly authorized to perform such actions, such as a multi-signature group. Similarly, metadata updates may be restricted to specific fields or actions based on asset type or business logic. This granular control is enforced through Algorand's smart contract logic, which acts as an immutable arbiter for all update operations.

The subsystem also incorporates a **transaction management component** to handle the complexity of blockchain interactions. When an update is initiated, the system generates a corresponding transaction payload using Algorand's JavaScript SDK. This payload includes details such as the NFT identifier, the type of update (e.g., metadata modification, transfer of ownership), and cryptographic signatures to validate the request. For multi-step updates, such as transferring ownership and modifying metadata simultaneously, the subsystem supports atomic transactions, ensuring that all operations succeed or fail as a single unit. To manage metadata updates, the design employs a hybrid approach that balances on-chain transparency with off-chain storage efficiency. Small or critical metadata fields, such as status flags or basic descriptions, are updated directly on the blockchain using Algorand's note field or asset configuration features. For larger or more complex metadata, the updated content is stored off-chain in decentralized storage systems like IPFS, with a content-addressable hash recorded on-chain. This ensures that updates are secure, immutable, and easily verifiable without overloading the blockchain. Additionally, the subsystem integrates with TRACE's **event logging and transparency framework**, recording every update transaction along with its details, such as timestamp, initiator, and change type. This enables a comprehensive audit trail for all NFT updates, which is essential for both compliance and user trust. The subsystem is also designed to handle concurrent update requests gracefully, ensuring that conflicting changes are resolved through priority rules or transactional locking mechanisms. The development of the NFT updating subsystem is grounded in best practices for blockchain programming, with a strong emphasis on security, integrity, and modularity. The system leverages Algorand's JavaScript SDK extensively to interact with the blockchain, constructing and signing transactions programmatically. For example, to modify an NFT's metadata, the code dynamically builds an asset configuration transaction, embedding the new data and ensuring compliance with ASAs.

Smart contracts play a pivotal role in enforcing update rules. Custom Algorand smart contracts (stateful or stateless) are written to validate and process specific update types. These contracts include logic to verify user signatures, check ownership, and ensure that updates conform to the NFT's predefined parameters. For instance, a smart contract might enforce that only authorized parties can update the "redeemable status" of an NFT linked to a physical asset, or it might require multi-signature approvals for transferring ownership of high-value assets. Security is paramount during coding. The codebase employs cryptographic techniques to verify user permissions, prevent unauthorized access, and ensure the integrity of updates. For example, before executing an update, the system verifies the cryptographic signature of the transaction payload against the public key of the authorized user. Replay attack prevention is also built into the system, with each transaction including unique identifiers and timestamps to ensure they cannot be reused maliciously. To support maintainability and scalability, the codebase is modular, with separate modules for metadata handling, transaction generation, and blockchain interaction. This allows for easier debugging, testing, and future enhancements. Additionally, robust error-handling routines ensure that failed transactions or unexpected issues do not compromise the system's stability or user experience. By adhering to these design and development principles, the NFT updating subsystem delivers a secure, efficient, and flexible solution for managing changes to digital assets within the TRACE ecosystem.

The NFT update subsystem is integrated with the blockchain and wallet subsystems to ensure that updates are executed correctly and tracked on the blockchain. Unit testing ensures that NFT updates (e.g., ownership changes, metadata alterations) function as expected. Integration testing ensures that updates are reflected correctly in the system, and security testing ensures that unauthorized changes cannot be made. The NFT updating subsystem is deployed alongside the NFT generation and blockchain interaction subsystems, ensuring seamless NFT lifecycle management. Post-deployment maintenance involves ensuring that the system remains secure and compatible with future upgrades to the Algorand network and the SDK.

D. Blockchain Interaction

The blockchain interaction subsystem serves as the critical bridge between the TRACE platform and the Algorand blockchain. This subsystem provides RESTful APIs that encapsulate blockchain functionality, enabling other TRACE modules to perform actions such as transaction execution, querying account details, and interacting with Algorand smart contracts. It is built to offer seamless access to

blockchain features while ensuring reliability, security, and scalability. The APIs are containerized using Docker for consistency and portability across environments and deployed using Jenkins to automate the integration and delivery pipelines. To simplify documentation, testing, and onboarding for developers, Swagger UI is integrated, providing an intuitive interface for exploring and interacting with the APIs.

The blockchain interaction subsystem must abstract the complexities of direct blockchain communication, making it easier for TRACE components to perform essential blockchain operations. Functional requirements include the ability to submit transactions, query blockchain states, and interact with Algorand smart contracts, all via a well-defined RESTful API layer. Security requirements include authentication mechanisms for API endpoints to prevent unauthorized access and ensure data integrity. Scalability is a critical non-functional requirement, as the subsystem must handle a growing number of concurrent requests without performance degradation. Other non-functional requirements include high availability, fault tolerance, and detailed logging for traceability.

The system architecture is designed to modularize blockchain interactions into reusable and scalable API endpoints. These endpoints are containerized using Docker to ensure a consistent runtime environment across different stages of development, testing, and production. Each API is stateless, promoting scalability by allowing multiple instances to handle requests concurrently. Jenkins is used to automate deployment pipelines, enabling rapid iteration and deployment of updates or new features. The architecture also includes an integrated Swagger UI, which serves as both a documentation tool and an interactive API testing interface. For blockchain connectivity, the subsystem leverages the Algorand JavaScript SDK to interact with the blockchain network, ensuring robust transaction management and data retrieval. A dedicated database (e.g., PostgreSQL) is used to cache frequently accessed blockchain data, such as account states or transaction logs, improving performance for read-heavy operations. Load balancers and auto-scaling mechanisms ensure that the system can handle high volumes of requests efficiently.

The blockchain interaction subsystem is divided into several components:

1. **API Gateway:** This handles incoming RESTful API requests, performs authentication and authorization checks, and routes requests to the appropriate service modules.

2. **Transaction Service:** Responsible for constructing, signing, and submitting transactions to the Algorand blockchain. It supports features like atomic transfers, group transactions, and smart contract calls.
3. **Query Service:** Facilitates reading operations such as fetching account balances, querying transaction details, or retrieving asset information.
4. **Caching Layer:** A middleware that stores frequently accessed blockchain data to reduce the load on the Algorand network and improve API response times.
5. **Logging and Monitoring:** Ensures that every API request and blockchain interaction is logged for debugging, analytics, and compliance purposes. Real-time monitoring tools (e.g., Prometheus, Grafana) are used for operational insights.

The subsystem is implemented using modern JavaScript frameworks such as Node.js to handle asynchronous requests efficiently. RESTful APIs are built with Express.js, ensuring lightweight and performant endpoints. Each API endpoint follows a clear and consistent structure, returning standardized JSON responses for ease of integration with other TRACE modules. The Algorand JavaScript SDK¹⁸ is used for blockchain interactions, ensuring compatibility with the Algorand protocol and supporting advanced features like atomic transfers. For enhanced security, JWT-based authentication is implemented for API access, ensuring only authorized users or services can interact with the endpoints. Swagger UI is integrated directly into the API gateway, enabling developers to view detailed API documentation and test endpoints without needing external tools. This enhances both development efficiency and collaboration. The RESTful APIs are containerized using Docker, ensuring they run consistently across different environments. Jenkins automates the CI/CD pipeline, handling tasks such as building Docker images, running tests, and deploying containers to production environments. API endpoints are integrated with the TRACE platform through well-documented interfaces, enabling seamless interactions between the blockchain and other subsystems like wallet integration, NFT management, and data analytics. Integration with Swagger UI provides real-time access to API definitions and testing capabilities, streamlining the process for developers integrating the API with other components.

The blockchain interaction subsystem undergoes rigorous testing to ensure functionality, performance, and reliability. Unit tests cover individual API endpoints, validating expected behavior under various scenarios. Integration tests verify that the APIs communicate correctly with the Algorand blockchain,

¹⁸ <https://algorand.github.io/js-algorand-sdk/>

ensuring end-to-end functionality. Load testing is performed to simulate high-concurrency scenarios, ensuring that the subsystem can handle the expected traffic volumes. Automated testing is incorporated into the Jenkins pipeline, enabling tests to be executed during every build and deployment cycle. Additionally, manual testing through Swagger UI ensures the accuracy of API responses and the integrity of blockchain interactions. Real-world scenarios, such as transaction failures, network timeouts, and invalid requests, are simulated to verify the subsystem's robustness and error-handling capabilities.

E. Data Management Module

A central component of the TRACE blockchain infrastructure is the **Data Management Module**, which plays a pivotal role in storing, managing, and ensuring the integrity of critical data. This includes records related to **dNFTs**, **transaction IDs**, and **logistics data** such as shipment statuses and delivery updates.

- **Data Integrity:** By using **cryptographic hashing** and secure storage protocols, the Data Management Module ensures that all records on the blockchain are immutable and tamper-proof. This guarantees the authenticity and reliability of the data throughout its lifecycle.
- **Scalability:** Given the growing volume of transactions and records, the system is designed to scale efficiently. This is achieved through a combination of decentralized storage solutions and **blockchain transaction optimization**, ensuring that the system can handle increasing demands without compromising performance or security.

F. Cloud Infrastructure: Nodely

The cloud infrastructure for TRACE, powered by Nodely, is a critical component that ensures scalability, high availability, and seamless interaction with both decentralized storage and real-time blockchain processing. The design of this infrastructure focuses on providing a reliable environment for the TRACE platform, handling large-scale data storage, rapid transaction processing, and high-performance computation necessary for blockchain interactions, particularly with the Algorand blockchain. This infrastructure is tightly coupled with decentralized storage solutions such as IPFS and Pinata, which are integrated to handle the large amounts of unstructured data (e.g., NFT metadata, digital asset files) associated with TRACE.

The cloud infrastructure must meet several key functional and non-functional requirements to support the decentralized nature of TRACE. Functional requirements include the ability to handle a significant number of concurrent transactions, store large volumes of metadata, and support seamless interaction with blockchain services like Algorand¹⁹. Additionally, it must provide access to decentralized storage solutions such as IPFS²⁰ and Pinata²¹, which will store NFTs' associated files (images, documents, videos) and metadata securely and in a distributed manner. Non-functional requirements focus on scalability, ensuring that the infrastructure can grow to accommodate increased transaction loads, larger amounts of data, and a growing user base. High availability is another critical non-functional requirement, as it ensures that TRACE services are always available and can recover quickly from potential failures. Performance metrics, including low latency for blockchain interactions and fast retrieval times for decentralized storage, are also crucial. Moreover, the cloud infrastructure must be secure to prevent unauthorized access and ensure data integrity, particularly for sensitive data stored off-chain.

TRACE's cloud infrastructure, built on Nodely, integrates decentralized and centralized services to provide a high-performance environment that can meet the demands of blockchain transactions and large-scale data storage. The system architecture consists of several layers:

1. **Decentralized Storage Layer:** This is where all NFT-related assets (images, videos, documents, and other metadata) are stored. IPFS and Pinata are key players in this layer, as they provide decentralized storage that ensures the permanence, redundancy, and availability of the data. Data is split into chunks and distributed across a global network of nodes, making it resilient to single-point failures while ensuring users can access the data quickly.
2. **Blockchain Layer:** At the core of the architecture is the Algorand blockchain, responsible for managing transactions, minting NFTs, and enforcing asset ownership and smart contract rules. This layer interacts directly with the cloud infrastructure through API gateways and microservices to execute blockchain transactions. Cloud services like Nodely provide the necessary computational power and ensure that these interactions can scale to handle the demands of TRACE's growing ecosystem.
3. **Transaction Layer:** This layer processes the transactions related to NFTs and digital assets. It handles the communication between TRACE's blockchain interactions and the decentralized

¹⁹ <https://arc.algorand.foundation/ARCs/arc-0019>

²⁰ <https://ipfs.tech/>

²¹ <https://docs.pinata.cloud/>

storage layer. When a new NFT is minted or updated, metadata related to the asset (such as ownership and attributes) is recorded on the blockchain, and the actual asset (e.g., image, video) is stored on decentralized storage like IPFS or Pinata.

4. **Data Processing and Query Layer:** This layer provides real-time data processing capabilities, including event logging, transaction tracking, and analytics. It integrates with the decentralized storage layer to retrieve and present asset metadata while ensuring that the data remains up to date. For example, if an NFT's metadata is updated, this layer ensures that the updated data is reflected both on-chain and off-chain.

The cloud infrastructure subsystem is designed with several key components:

1. **Microservices Architecture:** The system utilizes a microservices approach to break down the infrastructure into smaller, independent services, each with its own responsibility. For example, one microservice might handle interaction with IPFS or Pinata, while another microservice handles transactions on the Algorand blockchain. This modularity allows for easy scaling of each service as needed and enables flexibility in maintaining and upgrading specific components without affecting others.
2. **Storage Management Service:** This service manages all interactions with the decentralized storage layer (IPFS/Pinata). When an asset is uploaded, the service handles the communication with IPFS or Pinata, pins the file, and stores the reference hash in the blockchain or in a centralized database for easier retrieval. This ensures that all files linked to NFTs are permanently stored in a distributed manner.
3. **Blockchain Interaction Service:** This component communicates with the Algorand blockchain using the JavaScript SDK, sending and receiving transactions as needed. It coordinates with the wallet and NFT generation subsystems to perform transactions related to NFTs. It also verifies transactions, checks asset ownership, and interacts with smart contracts to validate conditions (such as approvals for transferring or updating NFTs).
4. **Caching Layer:** To optimize performance and reduce latency, a caching layer is used to store frequently accessed blockchain data or metadata. This layer can store data such as NFT ownership status or metadata hashes, reducing the need to query the blockchain or IPFS every time the information is required. This improves user experience by ensuring fast data retrieval.

The development of the cloud infrastructure subsystem focuses on building secure, scalable, and reliable services. Nodely, being a cloud-native platform, allows developers to create services that automatically scale based on demand. Each microservice is containerized using Docker, ensuring that the environment is consistent across development, testing, and production stages. Kubernetes is used to orchestrate the deployment of these containers, automatically scaling the number of instances based on demand, ensuring high availability and fault tolerance. The integration of IPFS and Pinata for decentralized storage requires developers to create secure and efficient APIs to handle file uploads, pinning, and retrieval. For each NFT minted or updated, the associated media is uploaded to IPFS or Pinata, and a reference hash is stored either on-chain or in a centralized database. Smart contracts are developed to ensure the integrity of these references and to govern access control rules, like ensuring only authorized users can update or transfer NFTs.

Cloud services are fully integrated into the overall TRACE ecosystem, ensuring scalability and high availability. For instance, blockchain interactions are seamlessly integrated into cloud-based services, enabling smooth transactions for asset minting and updates. Decentralized storage (IPFS/Pinata) is integrated with the cloud storage subsystem to ensure that NFT metadata and files are stored securely and are easily accessible. Real-time data processing services help keep users updated with the latest blockchain data and NFT changes, ensuring that the data presented across the platform is always current and consistent. The cloud infrastructure is tested for performance and scalability. Load balancing is tested by simulating large volumes of concurrent transactions and ensuring that the system can handle the increased load without degradation in performance. Stress testing is performed to validate the system's response to edge cases, such as server failure, network latency, or sudden spikes in traffic. Cloud-based systems are also tested for fault tolerance, ensuring that failures are handled gracefully, and users are not impacted by outages. The entire infrastructure is deployed in cloud environments using automated pipelines powered by Jenkins. The services are containerized, and each service is deployed as a Docker container on a cloud-native platform.

G. Decentralized Storage: IPFS and Pinata

One of the most innovative aspects of the TRACE blockchain system is its use of **decentralized storage**. Centralized systems struggle with high costs and vulnerability to data loss, but decentralized storage solutions like **IPFS** and **Pinata** address these challenges by enabling the storage and retrieval of data in a distributed manner across a network of nodes.

- **Content-Addressed Storage:** Files stored on **IPFS** are identified by their **content hashes**, meaning that files are always retrieved based on their unique content identifier (CID). This guarantees that the data being retrieved has not been altered or tampered with.
- **Reducing Blockchain Load:** By storing large files off-chain, such as shipment details, collaboration agreements, and product documentation, **IPFS** reduces the load on the blockchain. This ensures that the blockchain can focus on processing transactions and maintaining the ledger, while **IPFS** handles the heavy lifting of large data storage.

Pinata: Ensuring Availability and Reliability

While IPFS offers decentralized storage, it does not inherently ensure that the data remains available over time. To guarantee availability, TRACE utilizes **Pinata**, a service that acts as a **pinning service** for IPFS.

- **Pinning Files:** Pinata ensures that important data—such as **transaction logs** or **critical shipping information**—remains accessible by **pinning** it on the IPFS network. This ensures that even if no other nodes are hosting the file, the data remains available and retrievable.
- **Reliability:** By leveraging **Pinata**, TRACE ensures that its data is not only decentralized but also **highly available**, reducing the risk of data loss and ensuring the reliability of the system.

Data Availability and Performance

Using **IPFS** and **Pinata** in combination enables **TRACE** to achieve decentralized storage without compromising data **availability**. This also reduces costs associated with blockchain transactions by limiting the need for large-scale on-chain data storage. At the same time, the distributed nature of these technologies makes data more resilient to **server failures**, **downtime**, or **network attacks**, ensuring that TRACE's logistics operations are always running smoothly.

H. Event Logging and Transparency

A crucial feature of TRACE's blockchain infrastructure is **event logging**, which captures significant actions within the system, such as package pickups, deliveries, and status updates. These events are recorded directly onto the blockchain, ensuring that every action is transparent and immutable.

- **Transparency:** By logging these events directly on the blockchain, TRACE provides full transparency into the status and progress of shipments and transactions. This transparency is key in building trust with users and partners.
- **Immutability:** Once recorded, these events cannot be altered or erased, ensuring that the history of each asset is preserved for auditing and compliance purposes.

I. Algorand Testnet

The **Testnet** serves as a sandbox environment where developers can test the functionality of the TRACE blockchain system without interacting with real value. During the Testnet phase, all modules, including Algorand's blockchain and decentralized storage solutions (IPFS and Pinata), are thoroughly tested for performance, security, and scalability.

- **Testing Smart Contracts and dNFT Transactions:** During this phase, **smart contracts** deployed on the Algorand network are rigorously tested to ensure their functionality and compliance with the logistics use cases. **dNFTs**, for example, are tested to confirm that they correctly represent ownership, transactions, and status updates.
- **Simulating Real-World Operations:** TRACE's Testnet also simulates real-world logistical operations, such as the creation of shipments, tracking updates, and event logging, ensuring that these actions are correctly recorded and confirmed in the blockchain.
- **Security Audits and Penetration Testing:** The Testnet phase also includes rigorous **security audits** to identify any potential vulnerabilities within the system. This includes testing for smart contract exploits, **Sybil attacks**, and issues in the decentralized storage mechanism that could potentially compromise the integrity of the system.

- **Bug Fixes and Optimization:** As issues are discovered during testing, TRACE's developers optimize the system, improve transaction efficiency, and fix any identified bugs to ensure a seamless transition to the Mainnet.

By using the **Testnet** environment, TRACE ensures that all components are thoroughly vetted and ready for the scalability challenges of a real-world environment.

J. Transition to Mainnet

The transition from **Testnet** to **Mainnet** will be a critical milestone for TRACE. The **Mainnet** represents the fully operational blockchain, where all real-world transactions and activities occur. Though this activity is not planned for the duration of the TRACE project, the blockchain development is carried out keeping in mind the broader scope of mainnet deployment.

- **Mainnet Launch:** Upon transitioning to **Mainnet**, TRACE will begin conducting actual transactions, such as real-time shipment tracking, asset transfer, and dNFT updates. This will involve deploying **Algorand smart contracts** that will interact with both the blockchain and decentralized storage.
- **Scalability and Performance on Mainnet:** The **Algorand blockchain** is known for its **high scalability**, and this will be tested on Mainnet as TRACE handles a growing number of transactions and users. By leveraging Algorand's **Pure Proof-of-Stake (PPoS)**, TRACE will be able to maintain **low transaction costs** and **high throughput**, even as the system scales.
- **Security and Compliance:** Once on Mainnet, the blockchain infrastructure will maintain the same level of **security** as during Testnet, with enhanced monitoring and **compliance** tools to ensure that all transactions adhere to regulatory standards (AML, KYC, GDPR) for crypto payments in e-commerce. TRACE's integration with decentralized storage solutions such as **IPFS** and **Pinata** will be fully operational on Mainnet, ensuring that essential data like shipment details and contracts are securely stored off-chain but are still directly linked to the blockchain for integrity and transparency.

By gradually transitioning to Mainnet, TRACE ensures that it can handle real-world operations with full security and compliance, providing users with a **highly available, scalable, and cost-effective** solution.

5.4 Alignment with Project Architecture and Requirements

The following detailed explanation breaks down the alignment of TRACE's blockchain infrastructure, decentralized storage mechanisms, and system components with the overarching project architecture and operational requirements.

Blockchain Functionalities Designed for TRACE

The integration of a Public Key Infrastructure (PKI) system and a decentralized authentication layer should significantly enhance the security and trust within the TRACE platform. By leveraging PKI, the system ensures that all sensitive actions and data exchanges are authenticated using digital signatures, which guarantees both authenticity and non-repudiation. This decentralized authentication layer should allow users to securely access the platform using blockchain-based methods, such as wallet integrations, without compromising their private keys. The data governance mechanism should be robust, ensuring that all records are tamper-proof while maintaining the flexibility to update the state of the system when necessary. Importantly, each action should be immutably logged, preserving a complete history of changes for transparency and accountability. This way, TRACE can offer a secure, trustworthy environment where data integrity is preserved while providing the adaptability to reflect real-time updates or modifications in a verifiable manner.

- **Decentralized Identity Management:** Every stakeholder (e.g., logistics partners) should be assigned cryptographic key pairs to authenticate actions without relying on centralized authorities.
- **Digital Signatures:** All critical actions, such as shipment creation or dNFT updates, should be signed using private keys to ensure authenticity and non-repudiation.
- **Data Encryption:** Sensitive shipment data should be encrypted during transit and storage, using public keys to ensure it can only be decrypted by authorized parties.
- **Permissioned Blockchain Elements:** While Algorand is a public blockchain, certain permissions (e.g., who can initiate specific transactions) should be enforced through cryptographic policies to ensure secure role-based access.

5.4.1 Self Sovereign Identity (SSID) Module: (Blockchain Based Authentication System)

The SSID Module is a central part of TRACE's authentication system, using Public Key Infrastructure and blockchain-based wallets for secure, decentralized identity management. Instead of traditional username and password systems, users authenticate by signing cryptographic challenges with their private key, which remains on their device, removing the risk of credential theft. Each user is assigned a unique Algorand wallet as their digital identity, with the public address securely stored in the TRACE system for authentication. Integrated with TRACE's blockchain, wallet management, and transaction systems, the SSID Module ensures a simple yet secure user experience. It manages processes like registration, login, and session creation, using the Algorand blockchain and its JavaScript software development kit to connect with backend components for secure, decentralized authentication. Following is the detailed explanation of each step involved in the Stakeholder's authentication process;

A. Registration Process

The registration process creates the user's digital identity by generating an Algorand wallet, which serves as their public key identity on the TRACE platform.

API Development Flow:

1. User Initiates Registration:

- a. The frontend (e.g., React, Angular) sends a request to the backend API (/register endpoint).
- b. The backend, built in Node.js, handles the incoming HTTP request and triggers the wallet generation process using the **Algorand JavaScript SDK**.

2. Wallet Generation:

- a. Using the Algorand SDK, the backend will call the `algosdk.generateAccount()` function to generate a new wallet (a public/private key pair).
 - b. `generateAccount()` will return the wallet's private key (used for transaction signing) and the public key (address, which is used to uniquely identify the user on TRACE).
 - c. The private key is **never stored** in the backend; only the public address is stored securely in the TRACE database.
-

3. **Secure Key Storage:**

- a. The private key remains on the user’s device (typically in a secure storage or hardware wallet).
- b. Public keys are stored in the database in a hashed or encrypted form to avoid exposure.

4. **Response to User:**

- a. The backend sends the wallet's public key (address) to the frontend as a response. This public address is used for subsequent actions and serves as the user’s identifier in TRACE.

5. **Optional Metadata:**

- a. The user may add optional metadata during registration (e.g., name, email), which can be linked to the wallet address and stored in a decentralized storage solution like **IPFS** or **Pinata** for enhanced privacy.

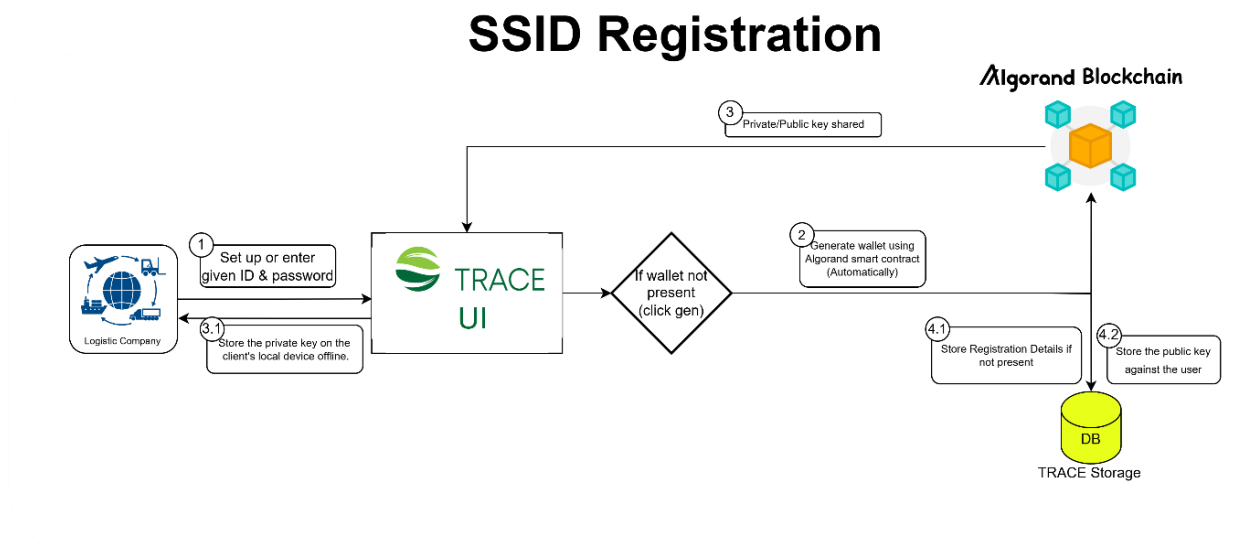


Figure 18: SSID Registration

B. Login Process

The login process verifies the identity of a user by matching the public key address stored during registration with the one provided during login and confirming the user's ownership of the private key via a signed challenge.

API Development Flow:**1. User Initiates Login:**

- a. The user submits their **public address** from their wallet via the front-end interface (e.g., login screen).
- b. A backend API call (`/login`) receives the public address and triggers the generation of a login challenge.

2. Challenge Generation:

- a. The backend uses the Algorand SDK and generates a **random challenge** string.
- b. This challenge will be unique for each login attempt and includes a **timestamp** and the user's public address. This ensures that the challenge cannot be reused or replayed.
- c. The challenge is stored temporarily in a **Redis** database or memory cache for quick retrieval during verification.

3. Challenge Sent to User's Wallet:

- a. The challenge data (including public address and timestamp) is sent to the frontend, where it is displayed to the user's wallet for signing.
- b. On the frontend, the user's wallet (e.g., a browser extension like **AlgoSigner**) signs the challenge with the user's **private key**.

C. Challenge Generation

This is the process of generating a challenge that is time-bound and uniquely linked to the login attempt. The challenge is generated by the backend API and needs to be signed by the user to prove their identity.

Tech Flow:**1. Generate Random Challenge:**

- a. A **UUID** or random string is generated, which is cryptographically unique. For example, using `crypto.randomBytes()` or `crypto.randomUUID()` in JavaScript.
- b. The challenge incorporates additional metadata, such as the current timestamp, the user's public address, and a nonce to prevent replay attacks.

2. Challenge Data Format:

- a. The challenge might look like this (in JSON format):

```
{
  "user_address": "ALGO_PUBLIC_ADDRESS",
  "timestamp": "CURRENT_TIMESTAMP",
  "challenge_id": "UNIQUE_CHALLENGE_ID",
  "nonce": "RANDOM_NONCE"
}
```
 - b. This challenge is serialized and sent to the frontend.
3. **Timestamp & Nonce:**
 - a. The backend ensures the challenge expires after a defined time period (e.g., 5 minutes). This is achieved by storing the timestamp and checking it during verification.

D. Challenge Signing:

The user signs the challenge with their private key in their wallet, confirming ownership of the wallet associated with the public address.

Tech Flow:

1. **Challenge Received by Wallet:**
 - a. Once the challenge data is sent to the frontend, the wallet (e.g., AlgoSigner) prompts the user to approve the signing of the challenge using their private key.
2. **Signing Process:**
 - a. On the wallet, the user's private key signs the challenge. In Algorand, this can be achieved using the `algosdk.signTransaction()` function if the challenge is wrapped in a transaction or a `sign()` function if it's just data.
3. **Returning Signed Challenge:**
 - a. The signed challenge is sent back to the backend via an API call (`/verify-challenge`).
 - b. The signature is typically returned in the form of a **base64** encoded string.

E. Challenge Verification

Once the challenge is signed, the backend needs to verify that the signature is valid, ensuring the user controls the private key associated with the public address.

Tech Flow:

1. Signature Validation:

- a. The backend retrieves the signed challenge from the user.
- b. Using the **Algorand SDK**, the system performs the following:

```
const verified =  
algorand.verifySignature(challengeData, signedChallenge,  
userPublicKey);
```
- c. The `verifySignature` method checks that the signature matches the challenge data and is signed by the corresponding private key.

2. Challenge Expiration Check:

- a. The backend checks that the challenge hasn't expired by comparing the stored timestamp against the current time. If expired, the login request is rejected, and the user is asked to generate a new challenge.

3. Successful Verification:

- a. If the signature is valid and the challenge is not expired, the user's identity is verified.

F. Verification Responses

Overview: After verifying the challenge, the system responds back to the user with the result of the authentication attempt.

Tech Flow:

1. Success Response:

- a. If the verification is successful, a **JWT (JSON Web Token)** or **session token** is generated for the user. This token contains claims (such as the user's public address and session expiration) and is stored in the browser's session storage or cookies.
-

- b. The backend returns an HTTP success status (200 OK) with the authentication token and user information.
2. **Failure Response:**
- a. If the verification fails, the backend responds with an error code (e.g., 401 Unauthorized) and a message indicating the failure reason (e.g., invalid signature or expired challenge).
 - b. The backend logs the failed attempt for security auditing.

Session Creation (Handled by TRACE): Session creation ensures that once a user is authenticated, they can perform actions within the TRACE platform without needing to re-authenticate for each request.

Tech Flow:

1. **Session Token Generation:**
 - a. After successful authentication, the backend creates a session token using **JWT**. This token is cryptographically signed to prevent tampering.
 - b. The JWT might contain:
 - i. user_address (public key of the user)
 - ii. issued_at (timestamp)
 - iii. expires_in (session expiration time)
2. **Storing Session Data:**
 - a. The session token is sent to the frontend where it's stored securely in **HTTP-only cookies** or **local storage**.
 - b. The backend also stores session metadata in a cache (such as **Redis**) for quick retrieval during future requests.
3. **Session Expiry Handling:**
 - a. The system automatically expires the session after a defined period of inactivity or a fixed duration.
 - b. On each request, the backend verifies the session token by decoding it and checking its validity (e.g., expiration, signature).

G. Tech Stack and APIs Used:

- **JavaScript** for backend development.
- **Node.js** for handling API requests and running the server.
- **Algorand JavaScript SDK (algosdk)** for blockchain interaction, wallet generation, and signature verification.
- **Express.js** for handling RESTful API calls.
- **Swagger UI** for API documentation and testing (to provide a user-friendly interface for interacting with APIs).

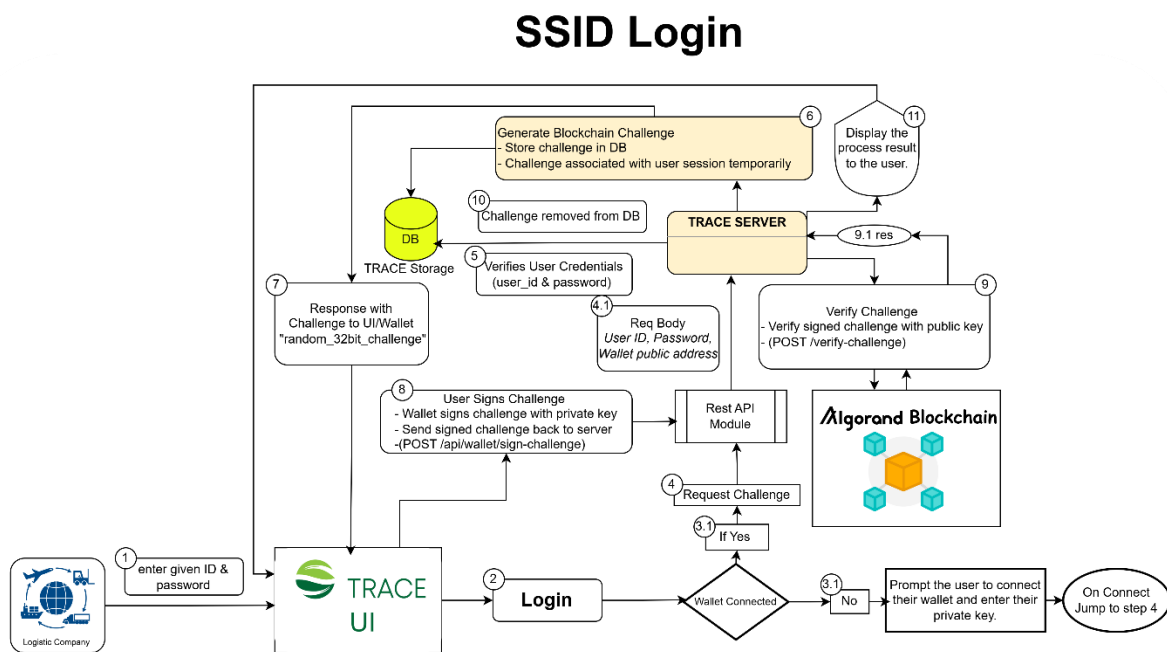


Figure 19: SSID Login

5.4.2 dNFT Module

The dNFT creation process is the cornerstone of TRACE's ability to digitize physical assets and represent them as dNFTs on the Algorand blockchain. This process establishes a seamless, secure link between the physical world and its digital representation, enabling enhanced traceability, ownership management, and asset lifecycle monitoring. The dNFT creation process incorporates a mix of blockchain

operations, metadata handling, and robust system integration, ensuring the highest levels of security, transparency, and compatibility.

A. dNFT Creation Process

The **dNFT creation process** in the TRACE system begins with the initialization of a **Dynamic Non-Fungible Token** on the Algorand blockchain using its **JavaScript SDK**. When a physical asset is registered in the TRACE ecosystem, its digital representation is created in the form of a dNFT. This process involves associating the asset with specific metadata, such as its origin, status, and owner. The creation process ensures the dNFT adheres to the **ASA** specifications, enabling compatibility and functionality with the broader Algorand ecosystem. Additionally, the creation process uses **smart contracts** to define update rules for the dNFT, ensuring only authorized operations can modify the token's metadata or state. Internally, an API call triggers the Algorand SDK to initialize the dNFT creation, sign the transaction using the user's wallet, and record the newly created token on the blockchain. This ensures tamper-proof documentation of the dNFT's initial creation and state.

B. dNFT ID Generation

The **dNFT ID generation** process involves creating a unique identifier for every dNFT, ensuring that it can be unambiguously recognized within the TRACE ecosystem. This ID is generated using a combination of the asset's details, such as a hash of its metadata, and a blockchain-specific identifier, ensuring both uniqueness and immutability. The ID generation process employs cryptographic hash functions, such as **SHA-256**, to prevent duplication and enable secure identification. This ID is linked directly to the dNFT on the Algorand blockchain, serving as the primary key for referencing the token. APIs expose this functionality for seamless integration into the TRACE system's workflow, ensuring all dNFT IDs are consistent and adhere to the system's standards.

C. Data Storage in IPFS

To handle asset-related data efficiently and securely, TRACE employs **IPFS** for decentralized data storage. Instead of storing metadata directly on-chain, TRACE stores large datasets (e.g., images, certificates) in IPFS, which provides a content-addressable storage system. This ensures scalability and data redundancy.

The process involves uploading the dNFT's metadata or associated files to IPFS via the **Pinata API**, which integrates seamlessly with IPFS. Once the data is uploaded, IPFS generates a unique **CID (Content Identifier)**, which is then linked to the dNFT on the Algorand blockchain. This architecture ensures data availability even in cases of system failures or network disruptions, while also reducing on-chain storage costs. APIs abstract the complexities of IPFS integration, enabling developers to upload, retrieve, and manage metadata effortlessly.

D. Wallet Requirements

The **Algorand wallet** is a fundamental requirement for dNFT operations. Every user interacting with the TRACE system must possess an Algorand-generated wallet, which provides the cryptographic keys necessary for signing transactions. The wallet ensures that all operations, such as dNFT creation, updates, or transfers, are securely authenticated and authorized. TRACE's wallet subsystem uses the **Algorand JavaScript SDK** to interact with wallets, managing public and private keys securely. Private keys are never stored on TRACE servers, ensuring complete decentralization and security. Multi-signature capabilities further enhance wallet operations, allowing collaborative approvals for high-value asset transfers or updates.

E. dNFT ID Mapping

dNFT ID mapping involves linking the unique dNFT ID with the physical asset it represents and the associated user wallet. This mapping is crucial for traceability and enables the TRACE system to maintain a clear relationship between on-chain tokens and off-chain assets. The ID mapping process integrates with TRACE's data management subsystem, storing mappings in a structured database for efficient querying and analysis. APIs handle operations for querying the mapping by dNFT ID, asset details, or wallet address, ensuring seamless data retrieval across the system. Blockchain calls verify the integrity of mappings, ensuring consistency between the database and the Algorand blockchain.

F. Metadata Updates

DNFTs require **metadata updates** to reflect changes in the state, location, or ownership of the associated physical asset. These updates are executed via **Algorand smart contracts**, which enforce rules for what can be updated and by whom. For example, ownership changes might require approval from both the sender and receiver wallets, ensuring secure transfers. The metadata update process involves calling

APIs that interact with the Algorand blockchain, using the JavaScript SDK to submit transactions. Updates are cryptographically signed by the authorized wallet and verified on-chain before they are committed. The system ensures data integrity and prevents unauthorized updates through strict access control and cryptographic verification.

G. Version Control

To maintain an auditable history of changes, the TRACE system implements **version control** for dNFT metadata. Each update creates a new version of the metadata while preserving the previous versions. This allows stakeholders to review the asset's lifecycle and ensures transparency in how the asset's details have evolved over time. Version control is managed through a combination of IPFS and blockchain. Metadata updates are stored in IPFS with a new CID generated for each version, and this CID is linked to the dNFT on the blockchain. TRACE's APIs expose version history retrieval, enabling users to query all versions of an asset's metadata efficiently. This architecture ensures that every change is recorded immutably and remains accessible to authorized users.

H. User Interface (UI) - Wallet Connection and Transaction Initiation

The **UI** of TRACE is the point of interaction for users, providing them with a seamless and intuitive method to connect their wallets, authenticate their identity, and initiate transactions. In TRACE, users are not required to use external wallet extensions like MetaMask or Trust Wallet. Instead, they connect through their **own Algorand-generated wallet**, which is a custom-built, secure solution integrated directly into the TRACE platform. Upon accessing the platform, users are prompted to **connect their wallet** via the UI. This simple step involves selecting the **Algorand wallet** they intend to use for authentication and transaction signing. Once the wallet is connected, the **UI** interacts with the **SSID Module** to verify the user's **public address**. This public address is checked against the data stored in TRACE, confirming that the user is authorized to access the system. After successful verification, the user is presented with a **cryptographic challenge** issued by TRACE. This challenge ensures that the person accessing the system is the rightful owner of the connected wallet. The challenge is a unique piece of data that must be signed by the user locally, using their private key. This process happens entirely on the user's device, ensuring that private keys never leave the wallet or device. Upon signing the challenge, the signed data is sent back to TRACE, which then verifies the signature. If the signature matches, the system creates a **secure session** tied to the user's wallet and public address. At this point, the user can proceed with various operations

within the TRACE system, such as interacting with **dNFTs**, initiating payments, or executing smart contract functions. The **UI** serves as the intuitive front end for these actions, guiding the user through the necessary steps and ensuring that all operations are securely authenticated and signed using the **Algorand wallet**.

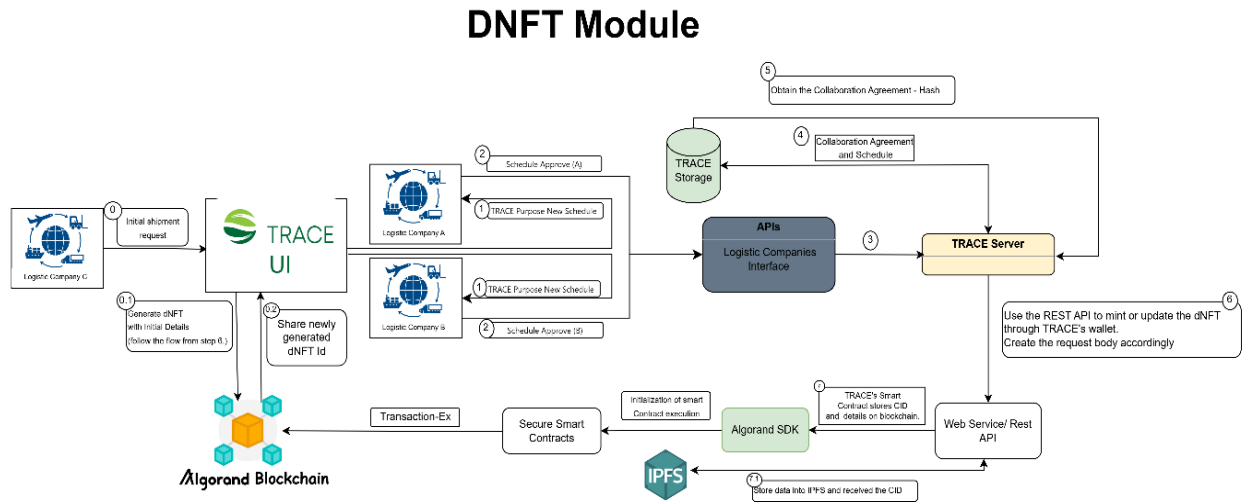


Figure 20: DNFT Module

5.4.3 Next Steps and Future Roadmap

We have already developed the fully functional independent SSID, DNFT and UI blockchain modules but as we look ahead to the next phases of the TRACE project, our focus will be on ensuring smooth integration with the TRACE ecosystem, transitions, scalability, and widespread adoption. Below are the key areas we will be concentrating on over the upcoming months and years:

A. Integration with TRACE and Tweaking as per Pilot Requirements

The immediate next step involves the **further integration of the TRACE system** with our operational framework. During this phase, we will continue refining the system based on the feedback and specific needs that arise from the **pilot regions**. The pilot phase is essential for gathering real-world data on the functionality and efficiency of the system. As we roll out TRACE in various pilot areas, including Greece, Italy, and Slovenia, we will identify and address any technical gaps or user-experience issues. This includes **optimizing the blockchain infrastructure**, **enhancing the user interface**, and ensuring that the data management and decentralized storage solutions meet the real-world demands of the logistics and

supply chain sectors. The tweaks and enhancements will be made in a controlled, iterative manner, ensuring that any modifications are in line with the system's core goals, such as **scalability, security, and transparency**. Additionally, we will be working closely with **stakeholders** in the pilot regions to ensure that regulatory and compliance standards are met. By incorporating pilot feedback into the system, we will ensure that TRACE is aligned with industry requirements and ready for future commercial deployment.

B. Transition to Mainnet and Commercial Deployment

Upon successful completion of the pilot phase and after addressing the feedback, the TRACE project will transition from the **testnet to the mainnet**. This transition is a crucial milestone as it marks the move from a testing environment to a fully operational, commercial-ready system. The transition will be executed in stages, ensuring that all components of the system—especially the **Algorand blockchain infrastructure** and **decentralized data management**—are robust enough to handle increased user activity and transaction volume. The mainnet deployment will occur **post-project duration**, and we plan for **commercial deployment on the mainnet within 36 months**. This extended timeline ensures that we have ample time for thorough testing, regulatory compliance, and system optimization. Once the mainnet is operational, TRACE will be available for **commercial use** by businesses and organizations that want to leverage its decentralized logistics and supply chain management capabilities. We anticipate that, once stabilized, the platform will be capable of handling a large volume of transactions, making it an ideal solution for **enterprise-level commercial deployment**. In this phase, the focus will be on ensuring that TRACE is ready for **mass adoption**, scaling efficiently to support commercial customers. The **Algorand blockchain**, with its high throughput and low transaction costs, will play a key role in ensuring that TRACE can handle these larger volumes without compromising performance or cost-efficiency. The **PKI authentication system** and **DNFT-based asset tracking** will be key differentiators that set TRACE apart from traditional logistics platforms.

C. Introduction of Tokenization

As the TRACE system stabilizes and becomes commercially viable, we plan to **introduce tokenization** as an additional layer to the platform. Tokenization refers to the process of creating digital tokens that represent real-world assets or values. In the context of TRACE, tokenization can enhance the system in several ways:

- **Supply Chain Tokenization:** Physical goods or shipments could be tokenized, with each **dNFT** representing a specific product or shipment within the TRACE ecosystem. These tokens could then be traded, sold, or transferred across different stakeholders in the logistics network.
- **Financial Tokenization:** TRACE could introduce its own **utility token** to facilitate transactions within the platform. This token could be used for paying transaction fees, accessing premium features, or incentivizing participation in the ecosystem. Additionally, businesses could utilize these tokens for **pay-for-performance** models or to reward trusted network participants, such as suppliers, shippers, or validators.
- **Creating a Tokenized Economy:** By incorporating tokenization into TRACE, we can create a **tokenized economy** where businesses can easily tokenize their assets, manage ownership, and transfer them within a secure, decentralized ecosystem. This would also provide the platform with a means to scale and facilitate deeper integration with broader blockchain ecosystems, such as **DeFi** protocols or **NFT marketplaces**.

The introduction of tokenization will not only add an additional revenue model to TRACE but also align the platform with the growing demand for **blockchain-enabled financial services**. It will make TRACE more flexible, enabling users to participate in a wider range of activities beyond simple supply chain management. Moreover, the use of tokenization will open up new opportunities for partnerships and collaborations, allowing TRACE to become a critical player in the **token economy** and **blockchain-based financial services**.

6 Implementation and Development

The Deliverable 4.3 focuses on the development of AI/ML and optimization algorithms for supporting Synchromodal logistics and shared resource optimization with emphasis on:

1. **Synchromodal Operations and Events Management (Task 4.3):** Delivering frameworks for events management in logistics networks.
2. **Logistics Optimization and Intelligent Scheduling (Task 4.4):** Enabling real-time data-driven decision-making.
3. **Blockchain Operations for Smart Contracts (Task 4.5):** More secure and automated transactions.

6.1 Methodology

The approach comprises the following components:

1. **Algorithmic Design and Implementation:**
 - AI/ML algorithms for the analysis of cargo, speed, position, and environmental factors.
 - Optimization algorithms for scheduling and resource allocation.
2. **Event Management:**
 - Frameworks to manage synchromodal logistics events across infrastructure and service levels.
3. **Blockchain for Secure Transactions:**
 - Decentralized smart contracts for transaction processing.

The tasks are designed to be interdependent:

- T4.3 provides event-related information to both scheduling algorithms (T4.4) and blockchain frameworks (T4.5).
- T4.4 Integrates data collected by IoT enabled logistics systems for improving freight scheduling and routing.
- T4.5 ensures secure data storage and immutable transaction records of shipments.

6.2 Phases and Milestones

Phase	Objective	Milestone	Date
Phase 1: Algorithm Development	Design and implement AI/ML algorithms for event management and logistics optimization.	AI modules ready	01/03/2025
Phase 2: Platform Integration	Integrate algorithms with TRACE platform, ensuring compatibility with infrastructural interfaces.	TRACE integration complete	01/04/2025
Phase 3: Blockchain Deployment	Implement and test blockchain-enabled transactions for logistics services.	Blockchain ledger operational	01/02/2025
Phase 4: Validation and Testing	Test the synchronodal operations and smart contracts in simulated and real-world environments.	Pilot evaluation complete	15/06/2025

6.3 Release Plan (M18 and Subsequent Releases)

M18 Release Objectives

1. Core Functionalities:

- AI/ML algorithms for event detection and management (T4.3).
- Initial implementation of the logistics scheduling module (T4.4).
- Functional blockchain modules for user authentication and for smart contracts (T4.5).

2. Expected Outcomes:

- Synchromodal operations aligned with TRACE architecture.
- Functionalities tested in controlled environments for readiness.

Subsequent Releases

- **M24 Release (Beta):**
 - Integration of additional optimization features, including environmental impact reduction.
 - Enhanced blockchain functionalities for real-time transaction tracking.
- **M36 Release (Final):**
 - Deployment in live logistics scenarios with all TRACE platform components fully integrated.
 - Comprehensive analysis of logistics efficiency improvements (fuel savings, emissions reduction, etc.).

7 Conclusion

The primary objective of this TRACE deliverable with title “Synchromodal operations and optimization of shared resources” was to present a comprehensive description of the progress and operational structures concerning the synchromodal logistics and resources optimization in TRACE project. In the previous sections were introduced methodology for the improvement of efficiency in operational using innovative technologies. This adoption of innovative technologies plays a critical role in the conception and implementation of applications for efficient resource management in collaborative logistic activities. Additionally, in this deliverable was investigated the role and the abilities of Synchromodal operations that are using in TRACE platform. Also, a transformation of traditional methods into optimized, technology-driven systems that applied in three scenarios of deliveries i.e., first, middle and last mile logistics were covered. Taking into consideration the trust that has to be established between the logistics partners and the TRACE project, the deliverable described the way that the blockchain technology is being used to maintain decentralized records of shipment data etc. The implementation of the components that are involved in the described procedure have been analyzed extensively in the respectively section.

8 References

- [1] B. Acero, M. J. Saenz and D. Luzzini, "Introducing synchromodality: One missing link between transportation and supply chain management," *Journal of Supply Chain Management*, vol. 58, no. 1, 2022.
- [2] J. Caceres-Cruz, P. Arias, D. Guimarans, D. Riera and A. A. Juan, Rich vehicle routing problem: Survey, vol. 47, 2014.
- [3] A. Kolinski, P. Nowak and M. Cudzilo, REVIEW OF INTELLIGENT SOLUTIONS TO OPTIMISE LOGISTICS PROCESSES AND IMPROVE EFFICIENCY, vol. 21, 2021.
- [4] J. K. Lenstra and A. H. Kan, Complexity of vehicle routing and scheduling problems, vol. 11, 1981.
- [5] C. S. Orloff, A fundamental problem in vehicle routing, vol. 4, 1974.
- [6] T. T. Doan, N. Bostel and M. H. Hà, The vehicle routing problem with relaxed priority rules, vol. 10, 2021.
- [7] M. M. Kaplan and K. P. Heaslip, "The Parallel Scheduling Vehicle Routing Problem for Multimodal Package Delivery*," in *2024 IEEE Intelligent Vehicles Symposium (IV)*, 2024.
- [8] M. Karkula, J. Duda and I. Skalna, "Comparisons of Capabilities of Recent Open-Source Tools for Solving Capacitated Vehicle Routing Problem with Time Windows," in *Carpathian Logistics Congress Proceedings CLC 2019, Zakopane*, 2019.
- [9] K. Tsolaki, T. Vafeiadis, A. Nizamis, D. Ioannidis and D. Tzovaras, Utilizing machine learning on freight transportation and logistics applications: A review, vol. 9, 2023.
- [10] P. De Petris, S. Khattak, M. Dharmadhikari, G. Waibel, H. Nguyen, M. Montenegro, N. Khedekar, K. Alexis and M. Hutter, Marsupial Walking-and-Flying Robotic Deployment for Collaborative Exploration of Unknown Environments, 2022.
- [11] H. Hourani, P. Wolters, E. Hauck and S. Jeschke, A marsupial relationship in robotics: A survey, vol. 7101 LNAI, 2011.

