TRACE

inTegration & haRmonizAtion of logistiCs opErations

D3.1 Report on reference architecture (A)

Horizon Innovation Actions | Project No. 101104278

Call HORIZON-CL5-2022-D6-02

Co-funded by the European Union

Dissemination level	Public (PU)
Type of deliverable	R – Document, report
Work package	WP3 – Platform Design and Integration
Status - version, date	Final v1.1, 20/10/2025
Deliverable leader	INTRA
Contractual date of delivery	30/11/2024
Actual date of delivery	30/11/2024

List of authors

Author Name	Organization
Themistoklis Anagnostopoulos, Konstantina Papachristopoulou	INTRA
Ioanna Mesogiti, Elina Theodoropoulou	OTE
Alessio Masola	UNIMORE
Theofilos Triommatis, Ioannis Papamichail	TUC
Anestis Papakotoulas, Vassilis Papataxiarchis	NKUA
Blaž Vukelić	AVLL
Panagiotis Kanellopoulos	ACS
Sheila Sánchez Rodriguez	ROB
Alexandros Dalkalitsis, Panagiotis Georgas	HT
Emanuele Zarfati	MOD
Nikolas Tymplalexis	UNISYSTEMS
Sareh Saeedi, Martin Sénéclauze	CSEM
Ines Pentek	UM
Kristijan Perčič	PS
Marco Gardini	DIFLY
Kolomvatsos Kostas, Kylafas Christos, Fountas Panagiotis	UTH

Version History

Version	Date	Author	Description of changes
0.1	16/07/2024	INTRA	Initial ToC
0.2	22/08/2024	INTRA	Initial input in Section 4.2
0.3	31/10/2024	OTE, UNIMORE, AVLL	Initial input in Section 5
0.4	01/11/2024	INTRA	List with Integration Points, Initial input in Section 1
0.5	19/11/2024	OTE, ACS, UNIMORE, ROBOTNIK	Finalisation of input in Section 5
0.6	27/11/2024	INTRA	Final edits, consolidation of inputs and creation of the final draft version for internal review
1.0	29/11/2024	INTRA	Creation of final version addressing internal review comments
1.01	22/05/2025	INTRA	Provision of template for addressing the revision requests
1.02	13/10/2025	TUC, NKUA	Additional input in Section 3 to address the request for revision
1.1	15/10/2025	INTRA	Creation of final revised version

Peer Review

	Reviewer Name	Organization	Date
		UNIS	
	Christian Chavez	ROB	29/11/2024

Quality Manager Review

	Reviewer Name	Organization	Date
	Ioannis Neokosmidis	INC	29/11/2024

Legal Disclaimer

Executive Summary

TRACE introduces an innovative platform designed to integrate services from diverse logistics operations through a comprehensive architectural framework. This report details the reference architecture that enables seamless integration of logistics services across multiple stakeholders and transportation modes.

The architecture is structured in interconnected layers, beginning with a user interface layer that provides intuitive access through APIs and graphical interfaces. This is supported by an application layer hosting core functionalities for scheduling, routing, and event management. The data management layer features the StreamHandler for real-time data processing and a Cloud-based Data Management System (CDMS) for persistent storage. At the foundation, the infrastructure layer supports various communication technologies including 4G/5G and WiFi, alongside diverse transport systems.

Platform interoperability is achieved through a sophisticated semantic framework that harmonizes data from different logistics sources. This is complemented by standardized integration points between components and a common TRACE data model enabling consistent data exchange. The architecture has been designed to support multiple demonstration scenarios, as evidenced by the Italian, Slovenian, and Greek use cases detailed in this report.

The platform incorporates several technological innovations, implementing a hybrid cloud-edge deployment for optimized performance and Vehicle-to-Vehicle (V2V) communication support. Security and transparency are ensured through blockchain integration, while real-time event detection and response capabilities enable dynamic adaptation to changing conditions. The architecture also provides comprehensive support for next-generation delivery methods, including autonomous vehicles and drones.

Through its design, the architecture enables efficient resource sharing and optimization while reducing operational costs and environmental impact. It enhances visibility and traceability throughout the logistics chain while supporting new business models. Security and data privacy are fundamental considerations, and the architecture facilitates seamless integration with existing systems through standardized interfaces and protocols.

This reference architecture serves as the foundation for implementing a scalable, secure, and efficient logistics platform that addresses the complex requirements of modern supply chain operations. The document provides comprehensive technical specifications for all stakeholders involved in the development, deployment, and operation of the TRACE ecosystem.

Table of Contents

Table of figures

Table of tables

Definitions, Acronyms and Abbreviations

Abbreviation	Definition
3GPP	3rd Generation Partnership Project
4G/ LTE	4th Generation Mobile Network – Long Term Evolution
5G	5th Generation Mobile Network
5G NSA	non-Stand Alone
5G SA	Stand Alone
5G/NG-RAN	5G/ New Generation - Radio Access Node
AAA	Authentication, Authorization and Accounting Server
AMF	Access and Mobility Management Function
AP	Access Point
API	Application Programming Interface
AUSF	Authentication Server Function
BSS	Basic Service Set
CDMS	Cloud-based Data Management System
DB	Database
DN	Data Network
dNFT	dynamic Non-Fungible Token
DS	Distribution System
DSM	Design Structure Matrix
EPC	Evolved Packet Core
ESS	Extended Service Set
E-UTRAN	Evolved-UMTS Radio Access Network
gPRC	Remote Procedure Calls
GPS	Global Positioning System
GW	Gateway
HSS	Home Subscriber Server
IAM	Identity and Access Management
IDM	Intrusion Detection Module
IEEE	Institute of Electrical and Electronics Engineers

Abbreviation	Definition
IoT	Internet of Things
ISP	Internet Service Provider
JSON	JavaScript Object Notation
MAC	Media Access Control
MME	Mobility Management Entity
NF	Network Function
NRF	Network Repository Function
NSSF	Network Slice Selection Function
OFDMA	Orthogonal Frequency-Division Multiple Access
PaaS	Platform-as-a-Service
PCF	Policy Control function
PCRF	Policy and Charging Rules Function
PGW	Packet Data Network Gateway
PKI	Public Key Infrastructure
PLMN	Public Land Mobile Network
QoS	Quality of Service
RAN	Radio Access Network
RDBMS	Relational Data Base Management System
REST	Representational State Transfer
SCP	Service Communication Proxy
SGW	Serving Gateway
SMF	Session Management Function
SQL	Structured Query Language
TDMA	Time-Division Multiple-Access
TMS	Traffic Management System
TW ADR	Twenswheel Autonomous Delivery Robots
TWT	Target Wake Time
UAV	Unmanned Aerial Vehicle
UDM	Unified Data Management

Abbreviation	Definition
UDR	Unified Data Repository
UGV	Unmanned Ground Vehicle
UI	User Interface
UML	Unified Modeling Language
UPF	User Plane Function
V2V	Vehicle-to-Vehicle
VPN	Virtual Private Network
WMS	Weather Management System

1 Introduction

1.1 Scope of the Deliverable

Deliverable D3.1 entitled "Report on reference architecture (A)" has been prepared in the framework of WP3 "Platform Design and Integration" and reports on the specification of the TRACE architecture, providing information on the details of integration, API formalization and the deployment guidelines. This is the first part of two releases, with D3.2 being the second and final release at M32 (after the Amendment to the Grant Agreement takes effect).

Deliverable D3.1 is connected to multiple tasks, and therefore presents the work performed and the results achieved in:

- T3.1 Holistic Platform Specification
- T3.2 Data Sharing Protocols and Interfaces
- T3.3 Big Data Management Reference Architecture
- T3.4 Transport and Communication Infrastructures

1.2 Relation with other Work Packages/Deliverables

Based on the Description of the Action (DoA), D3.1 has several relationships with other Work Packages and deliverables. In particular, it builds upon the technical requirements described in D2.1 and the ecosystem development and use cases presented in D2.4, both prepared in the framework of WP2 "Conceptual Framework". Additionally, D3.1 informs the deliverables of WP4 "Infrastructure & Ecosystem" referring to the infrastructural elements (D4.1, D4.2) and the synchromodal operations (D4.3, D4.4). Since D3.1 is part of a series with D3.2 (Report on reference architecture B), there is a direct dependency. Also, there is a strong connection with the three platform releases, namely D3.3 (alpha), D3.4 (beta), and D3.5 (final).

1.3 Intended Audience

Since this is an architecture specification document, the primary audience would be technical stakeholders, while still being accessible and valuable to other stakeholders who need to understand the system's overall structure and capabilities. Such technical stakeholders include software developers and system architects, technical partners within the consortium, integration specialists and infrastructure engineers, as well as other technical projects working on logistics platforms.

The dissemination level of the present report is Public, which means the deliverable is fully open and will be automatically posted online, making it accessible to all audiences. In this respect efforts have been placed for the document to be written in a way that balances technical depth with sufficient context and explanation, making it useful to this broad audience range.

1.4 Deliverable Structure

The report is structured in a way that presents the TRACE Architecture and its components, workflows and interfaces in a meaningful way.

Section 1 provides the introduction, mentioning the scope of the deliverable, including the position of the report in the report with a mention to its association with other deliverables and with the work performed in other Work Packages. Also, presents what the intended audience is.

In **Section 2**, the overview of the TRACE Architecture is presented, elaborating on the methodology followed. In addition, the platform workflows are explained. A detailed diagram depicts the components of the platform, which are then presented in more detail. Finally, this section concludes with the description of the interfaces between the platform and all its components. The contents of this section give an overview of the work performed in T3.1.

The interoperability layer is presented in **Section 3**, as per the work of T3.2. TRACE interoperability layer enables seamless data integration across different logistics databases through a unified semantic framework, allowing diverse systems to communicate using a common language without requiring data replication.

Section 4 describes the two elements of the big data management system, namely the StreamHandler platform and the Cloud-based Big Data Management System, which have been developed in the framework of T3.3.

Section 5 gives an overview of the transport and communication infrastructures, set in T3.4. The Transport and Communication Infrastructure layer provides the foundational hardware and connectivity needed for TRACE operations, incorporating various transportation assets (vehicles, drones, robots) and network technologies (4G/5G, WiFi, V2V) across Italian, Slovenian, and Greek demonstration sites.

The report concludes with **Section 6**, where the overall outcome of the work performed is briefly summarised, including mentions of future work and next steps.

1.5 Changes in relation to v1.0

This revised version of D3.1 (v1.1) has been developed in response to feedback received during the mid-term review of the TRACE project, that took place on February 4th, 2025, and according to the following request for revision:

"*The Interoperability Layer should be elaborated further. The revised document should include specific integration pathways for TRACE's compatibility with existing TMS/WMS systems and middleware solutions*."

The primary enhancement addresses the reviewer's recommendation to elaborate further on the Interoperability Layer, specifically requiring the inclusion of detailed integration pathways for TRACE's compatibility with existing Traffic Management Systems (TMS), Weather Management Systems (WMS), and middleware solutions. To address this requirement, a new section (*Section 3.5 Integration Pathways with External Traffic and Weather Management Systems*) has been added, which provides the technical specifications for connecting TRACE components with such external data sources.

2 TRACE Architecture

This Section describes the TRACE reference architecture, which provides a blueprint for system components and their interactions. It ensures that all parts of the system work together seamlessly, supporting scalability, maintainability, and flexibility to create a robust software platform that meets the needs of diverse stakeholders and supports the integration of various logistics services and technologies.

This section consists of the following subsections: Section 2.1 describes the methodology the consortium followed for the formulation of the TRACE Architecture; Section 2.2 reports the identified Platform workflows; Section 2.3 presents the high-level architectural overview diagram of the TRACE Platform, including high-level information about the included components; Section 2.4 presents the Platform components diagram; and finally, Section 2.5 provides an analysis of the Platform interfaces.

2.1 Methodology

Designing the architecture for a complex and modular platform like TRACE requires a structured and iterative methodology, making sure that the various stakeholder requirements will be fulfilled. From gathering initial requirements to defining high-level concepts, detailing components, and planning deployment, this approach provides a clear framework for creating a robust, flexible, and future-ready platform. Figure 1 presents the methodology that the consortium adopted, and the fundamental steps involved in translating stakeholder needs into a fully deployable system architecture. By leveraging modern design principles such as microservices and event-driven architectures, alongside best practices for implementation and validation, this methodology ensures the TRACE platform is optimised for real-world operations and seamless integration with external systems.

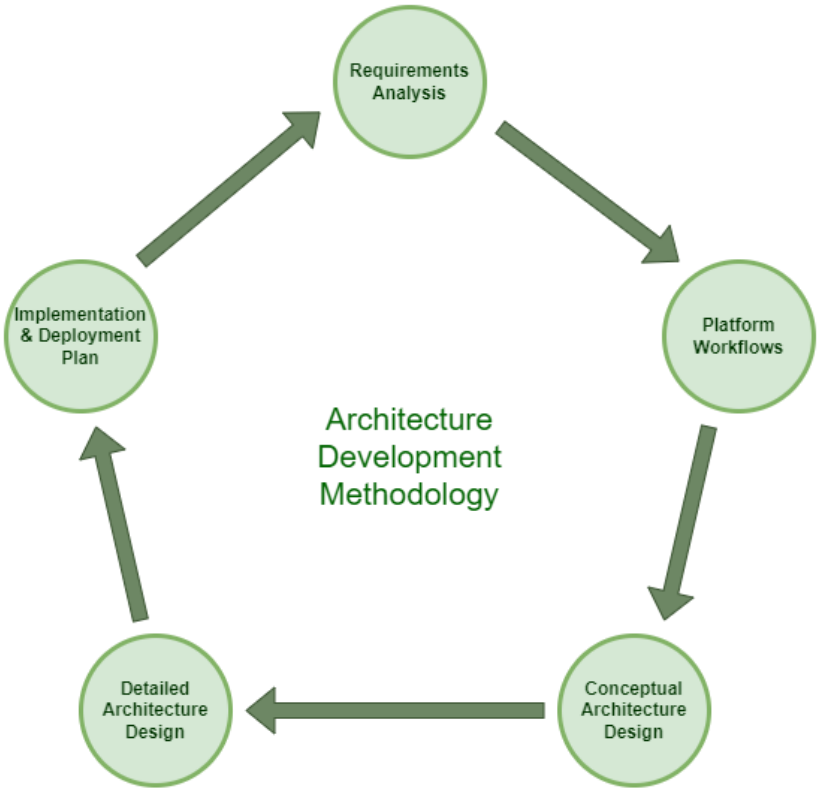

Figure 1: Architecture Development Methodology

The following subsections describe the different phases comprising the methodology.

2.1.1 Requirements Analysis

The requirement analysis is a fundamental phase that identifies the goals and operational challenges of the stakeholders, as well as their expectations from the platform. The main part of this exercise was conducted in the context of WP2. Through workshops, questionnaires, and interviews, the consortium identified key functional requirements, such as real-time data collection, route optimisation, and scheduling, as well as non-functional requirements, such as scalability, performance thresholds, and security compliance (e.g., GDPR). These requirements are categorised based on importance using prioritisation techniques. D2.1 [1] provides a comprehensive list of the identified requirements, as well as a detailed description of the process applied to collect and document these requirements. Each

requirement was then mapped to user stories that illustrate how stakeholders will interact with the platform, leading to the next phase of the methodology.

2.1.2 Platform Workflows

This phase focuses on translating stakeholder needs into actionable workflows that define how the TRACE platform will operate in real-world scenarios. Each workflow investigates how users and services interact with the platform and the expected outcomes. This phase started by drafting some high-level user stories. Then, based on these user stories, the technical teams developed the platform workflows, specifying the sequence of interactions between the end users, platform components, and external systems. The produced sequence diagrams that visualise the platform workflows are described in detail in Section 2.2. This step ensures that the user requirements and needs are aligned with the system's capabilities and functionalities.

2.1.3 Conceptual Architecture Design

The main goal of this phase is to define a high-level structure of the TRACE platform that includes all the functional blocks of the system. These blocks are organised into horizontal layers, such as User Interface, Application, Data Management, and Infrastructure, with each layer assigned specific responsibilities. Additionally, it contains the domains that the platform covers. The result is a high-level architecture diagram that visualises the platform components, layers, and domains, providing a foundation for the detailed design phase. This diagram is presented in detail in Section 2.1.2.

2.1.4 Detailed Architecture Design

Building on the conceptual design, this phase involves specifying the technical details and interconnections of each element. Every platform component was further analysed in terms of internal modules, tools, and services. Additionally, its inputs, outputs, dependencies and technologies were documented. As a starting point, the technical teams of the consortium created a list with all the platform components and their internal modules (the list is available in *ANNEX A: TRACE Platform Components List*). Subsequently, the integration points between components were detailed, specifying API endpoints, communication protocols (e.g., REST or gRPC), and data schemas. This phase concludes with i) a comprehensive architecture diagram depicting all the components' deployment and interfaces (Section 2.4) and ii) a table documenting all their integration points (Section 2.5).

2.1.5 Implementation and Deployment Plan

The final phase involves creating a concrete implementation, integration, and deployment plan for the whole platform. The TRACE technical teams followed a CI/CD methodology combined with some fundamental Agile principles to break down the development, testing, integration, and deployment of the various services into smaller, manageable sprints. CI/CD pipelines are set up to automate building, testing, and deploying microservices, ensuring smooth integration across components. The roadmap of the platform implementation and deployment is presented and analysed in D3.3 [2].

2.2 TRACE Platform Dataflows

To better understand the TRACE platform functionalities and interactions, the core technical teams of the project formed three main platform dataflows to describe different end-to-end use case scenarios that involve all of the platform components. These dataflows considered the platform conceptual architecture described in the previous section, the user and technical requirements produced in WP2, and the technical specifications of the involved components. The following subsections present these dataflows with a detailed sequence diagram and a short description.

2.2.1 Routing and Scheduling

This sequence diagram illustrated below showcases the process flow for managing shipment route requests between various components and stakeholders, such as two Logistics Companies. This end-to-end use-case scenario starts from a Logistics Company's shipment request and involves multiple components that interact to ensure an optimised and secure flow of the shipment.

The key platform components involved in this dataflow are:

- **User Interface:** Manages user interactions for schedule requests, updates, and approvals.
- **Scheduler:** Coordinates scheduling, optimisation, and finalisation of shipment routes.
- **Route Optimizer:** Retrieves data and generates optimised routes.
- **Blockchain:** Manages creation and updates of digital assets (dNFT).
- **CDMS:** Data management services.
- **Interoperability Layer:** Interfaces with external logistics data sources.

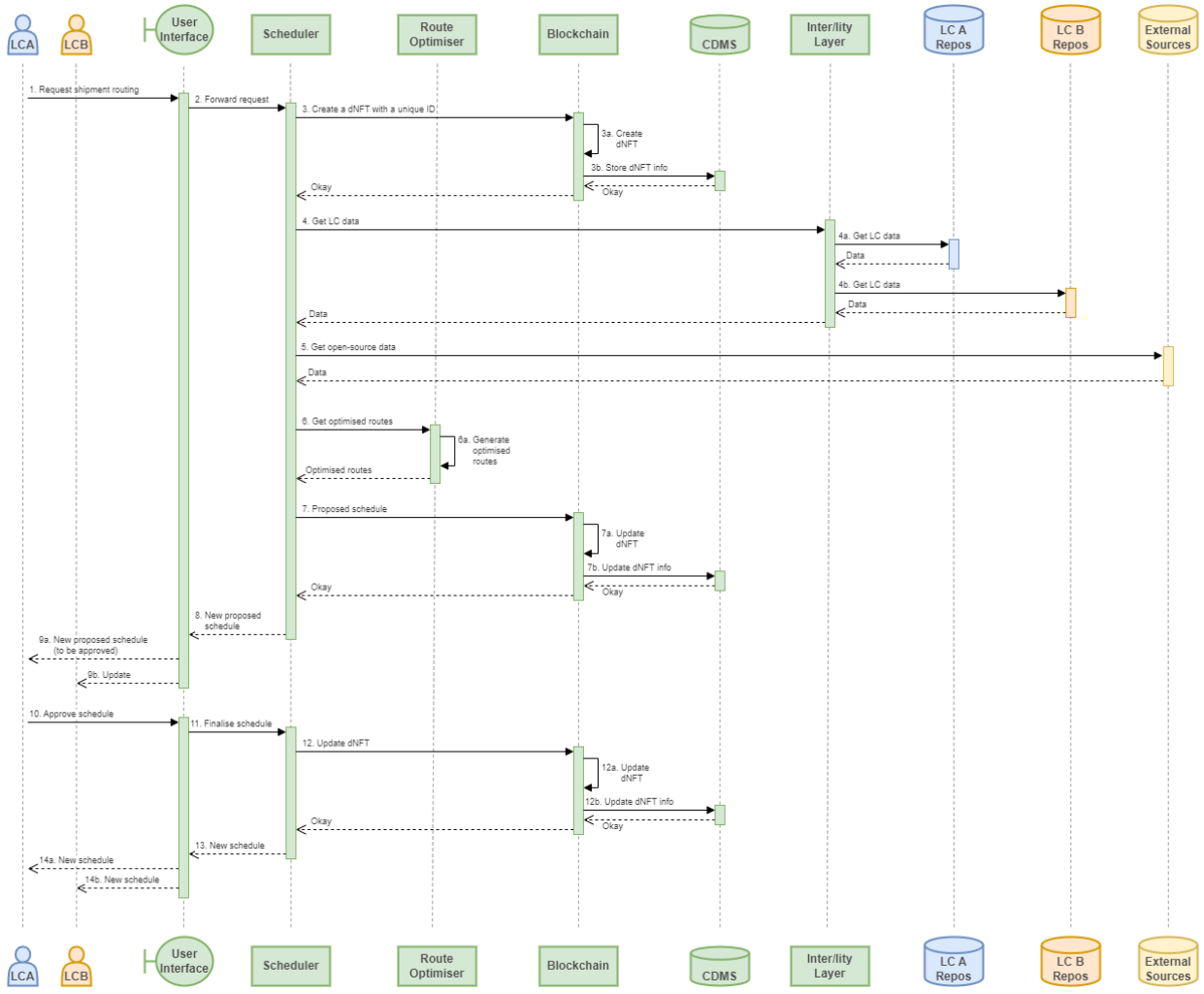

Figure 2: TRACE Routing Sequence Diagram

The involved steps are the following:

1. Shipment Route Request Initiation: A logistics company initiates a request for a shipment route through the User Interface.
2. Forwarding the Request: The User Interface forwards the received request to the Scheduler.
3. Store shipment-related data: The Scheduler interacts with the blockchain APIs to generate and store a unique shipment reference (dNFT).
4. Get Logistics Companies' data: The Scheduler uses the Interoperability layer to retrieve the available resources from the Logistics Companies.

5. Get Open-Source data: The Scheduler retrieves various types of data (e.g., weather updates, traffic conditions, reported events, etc.) from various open-source sources.
6. Route Creation and Optimisation: The Scheduler sends a request to the Route Optimiser, which uses the data to generate optimised routes. It then returns the optimised routes back to the Scheduler.
7. The Scheduler updates the dNFT information through the Blockchain APIs to include the proposed shipment schedule.
8. The Scheduler returns the proposed shipment schedule to the user interface.
9. The User Interface will send the proposed shipment schedule to the Logistics Company that requested the shipment. Additionally, it sends a notification to the other logistics companies, informing them of the routes they are involved in in the proposed schedule.
10. The requester approves the proposed shipment through the system User Interface.
11. The User Interface instructs the Scheduler to finalise the schedule.
12. The Scheduler finalises the schedule and updates the associated dNFT information on the Blockchain.
13. The Scheduler sends the finalised schedule to the User Interface.
14. The finalised schedule is shared with LCA/LCB through the User Interface.

2.2.2 Sensor-based Detected Event

This sequence diagram shows the procedure to be followed when an unexpected event occurs. Through the sensors and the mechanisms that monitor the data coming from them, TRACE can detect an event through rule-based mechanisms and then forward to the Scheduler the necessary data to reroute a process.

The key components that are involved in Detected Event are the followings:

- **StreamHandler:** Gathers and streams sensor data to other components.
- **Fleet Monitoring Manager:** Monitors vehicle availability and continuously streams vehicle data.
- **Monitoring Module:** Forwards data for event detection, tracks any deviations from the transfer plans delivered by Scheduler or Route Optimizer.
- **Event Management Module:** Receives input from Monitoring and makes a decision if an event is present.

- **Mitigation Manager:** Provides appropriate mitigation plans based on detected events.
- **Scheduler:** Gathers all the needed information from the components and if an event is present reschedules and reroutes a new plan.

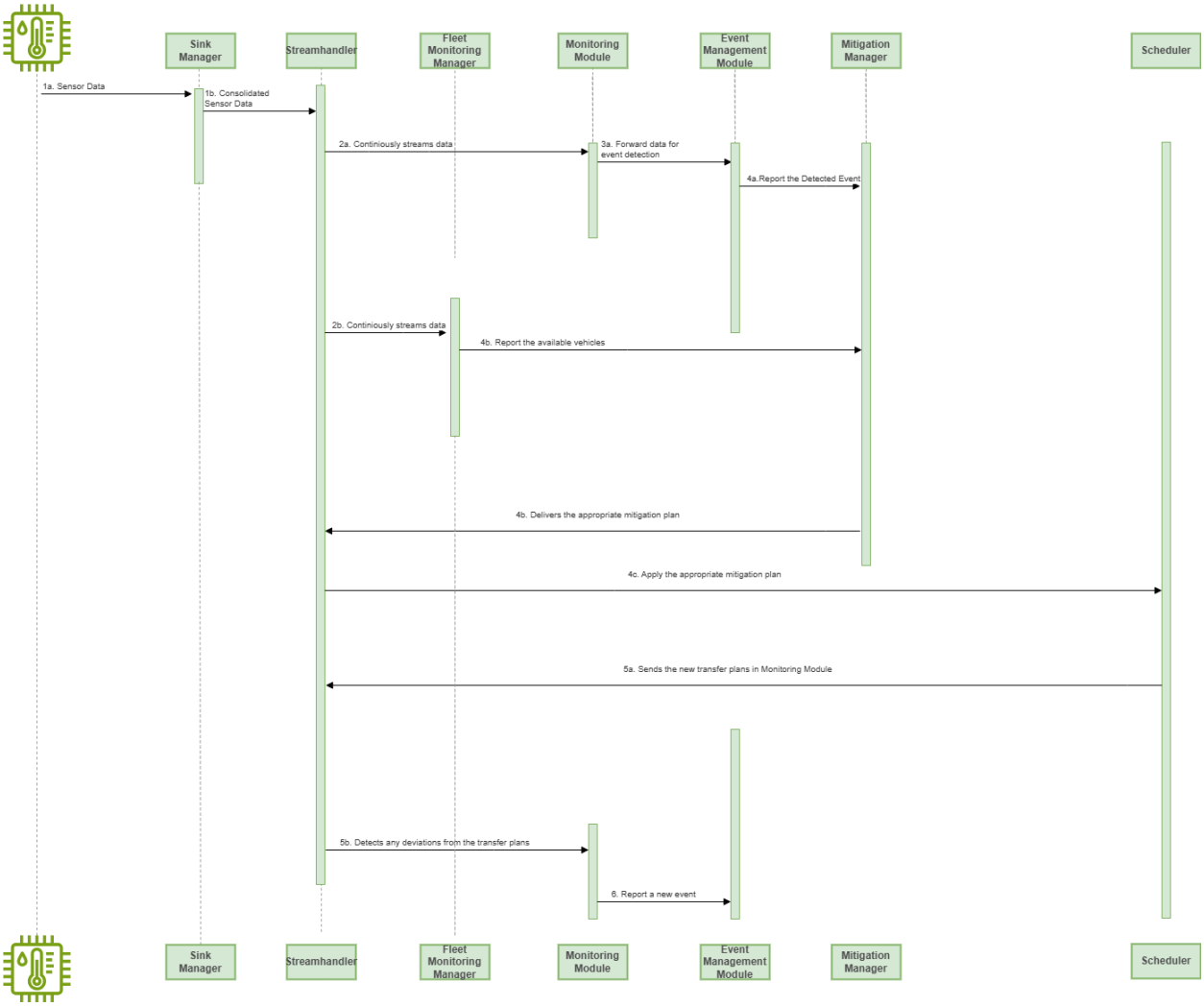

Figure 3: TRACE Event Sequence Diagram

A brief analyzation of the steps is:

1. The StreamHandler starts by gathering sensor data and streaming it to the Fleet Monitoring Manager.
2. The Fleet Monitoring Manager continuously streams data, providing real-time updates on vehicle statuses.

3. The Monitoring Module receives information from the Fleet Monitoring Manager and sends the information for event detection.
4. The Event Management Module may detect and report an event to the Mitigation Manager, if any.
5. Fleet Monitoring Manager supplies available vehicle information to the Mitigation Manager.
6. The Mitigation Manager provides the appropriate mitigation plan based on the detected event and the available vehicles.
7. The Mitigation Module takes the received mitigation plan and executes it if required.
8. The Mitigation Manager then sends any updated or new transfer plans back to the Monitoring Module.
9. The Monitoring Module detects any deviation from the transfer plans.
10. In case the deviation is found, then a new event is sent to the Event Management Module for further actions.

2.2.3 Last Mile

This sequence diagram shows the process that the TRACE platform follows during a last mile delivery. Through the logistic companies in cooperation with the companies that have autonomous vehicles the last mile delivery is always controlled by the Human Operator. While there is also the interaction between the unmanned vehicle and the customer.

The key components that are involved in Last Mile are the followings:

- **ACS ICT System:** It originates requests for shipment allocation and updates shipment status to delivered.
- **TRACE Platform:** Coordinates shipment allocation, calculates delivery routes, and shipment advancement.
- **Unmanned Aerial Vehicles:** Controls air parts of the shipment, location, and speed, and updates the status.
- **Unmanned Ground Vehicle:** Controls the land part of the shipment for appropriate delivery.
- **Operator:** Manages shipment pickup at the exchange location and assists with customer interaction.
- **Customer:** The last recipient of the delivery communicates with the operator to take the parcel.

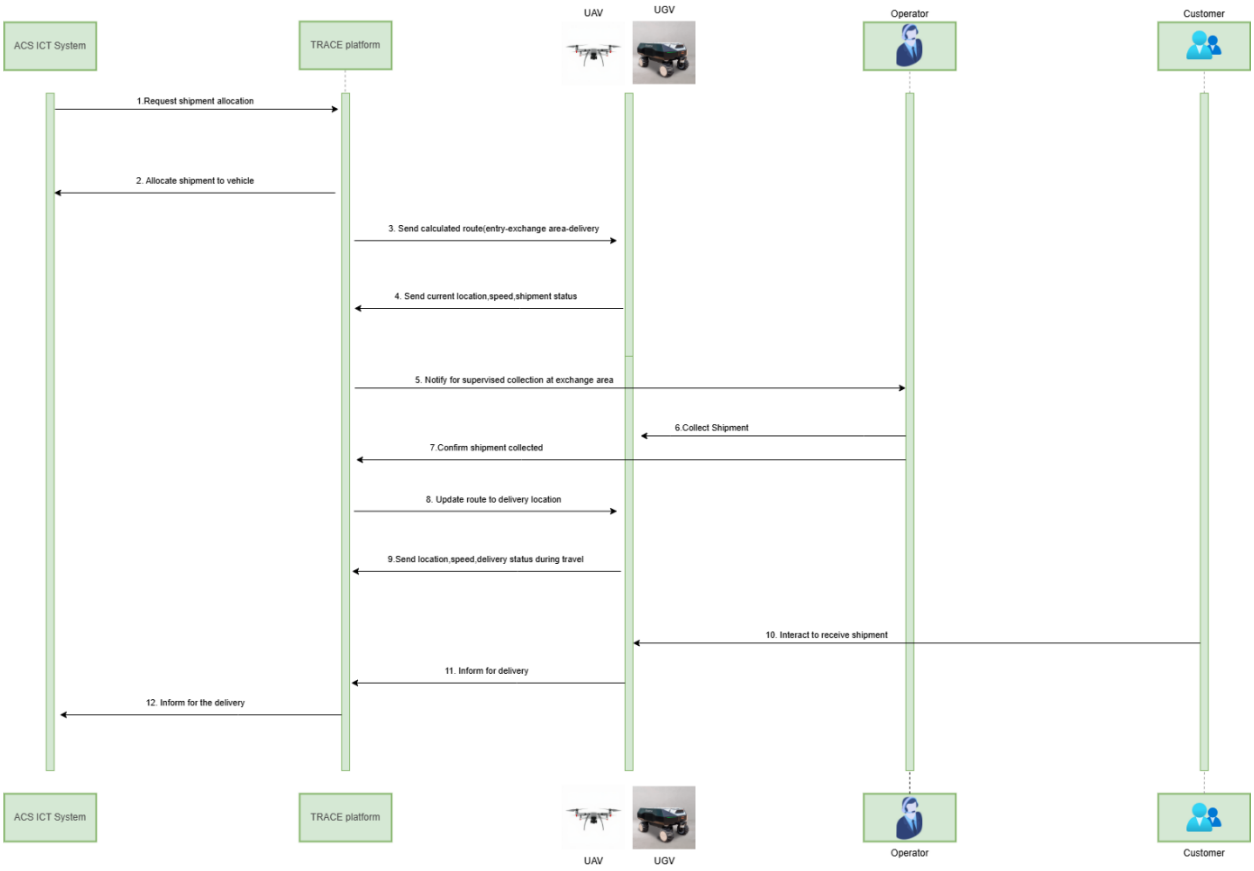

Figure 4: TRACE Last Mile Sequence Diagram

A brief analyzation of the steps is:

1. Shipment allocation request: The ACS ICT System raises a request to the TRACE platform for the allocation of a shipment to a vehicle.
2. Shipment allocation: The TRACE platform performs the allocation of the shipment to the appropriate UAV or UGV.
3. Route calculation and assignment: The TRACE platform sends the calculated delivery route including entry, exchange area and delivery back to the UAV/UGV.
4. Status updates during transit: The UAV/UGV sends updates about its current location, speed and shipment status to the TRACE Platform until they reach the exchange area.
5. Notification of supervised collection: At the exchange area the UAV/UGV notifies the operator for supervised collection.
6. Shipment collection: The operator collects the shipment at the exchange area.

7. Confirm shipment collection: The operator confirms the shipment has been collected and informs the TRACE platform.
8. Route update for final delivery: The TRACE platform updates the route to the final delivery location.
9. Ongoing status updates during final transit: The UGV continues with sending the updates on location, speed and delivery status.
10. Customer interaction and shipment handover: The customer interacts with the unmanned vehicles to hand over the shipment.
11. Confirm delivery: TRACE informed for the delivery by UGV/UAV.
12. Shipment status update: The ACS ICT system is informed for the delivery by TRACE.

2.3 TRACE Conceptual Architecture Design

The architecture diagram in Figure 5 represents the high-level structure of the TRACE platform, illustrating how different components are identified and organised into distinct layers and domains. The architecture follows a highly modular approach, where multiple TRACE components work together as microservices to optimise its functionalities related to transportation, logistics, data management, and communication infrastructure.

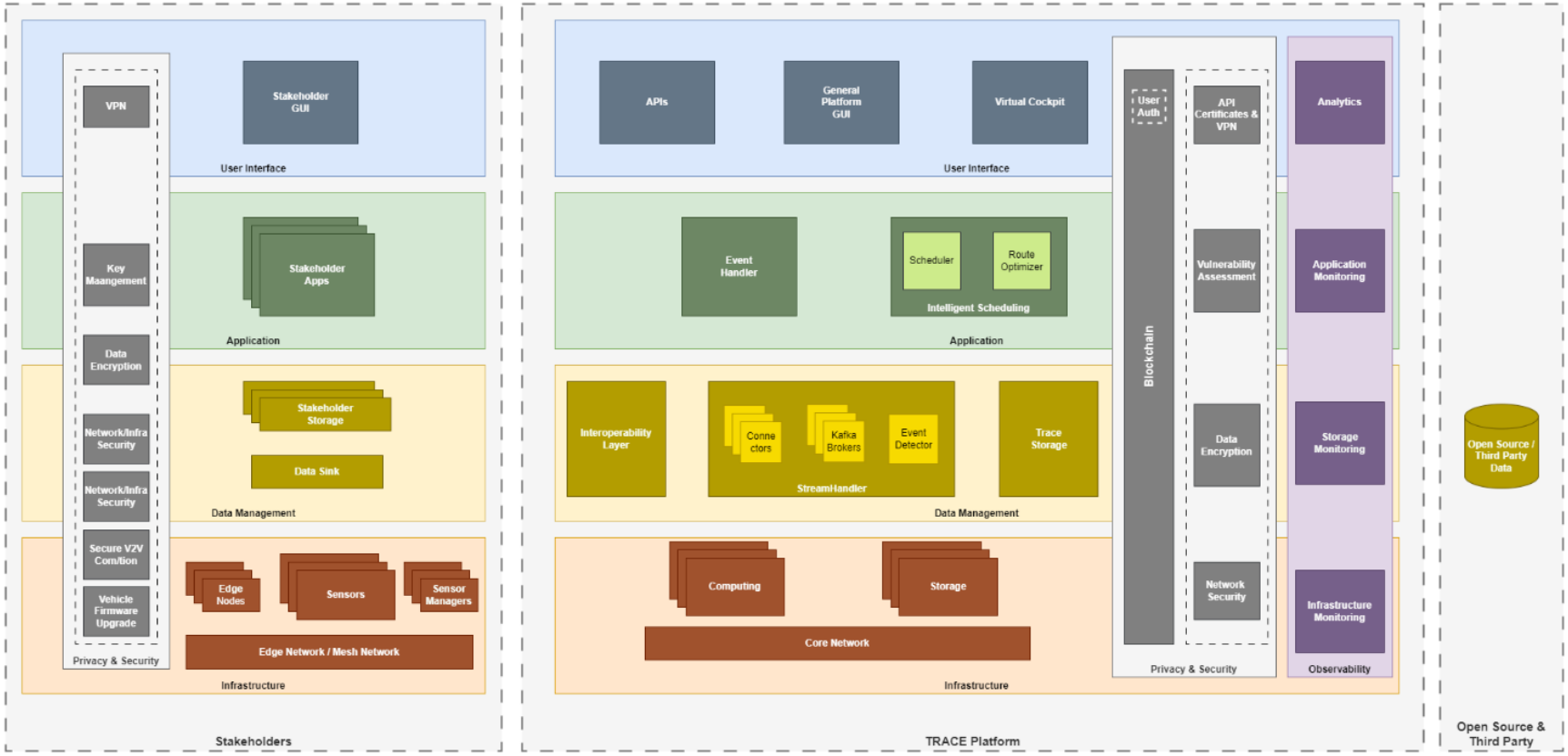

Figure 5: TRACE Platform Conceptual Architecture

As the diagram shows, the platform components are divided into horizontal layers. The components in each layer utilise other components from the same layer, or the layers beneath that, to offer a concrete service and functionality. From bottom to top, these layers are: i) infrastructure, ii) data management, iii) application, and iv) user interface.

These layers extend into three domains that are involved in the TRACE platform operations: i) the Stakeholders' domain, ii) the TRACE Cloud Platform, and iii) the Open-Source Domain. The Stakeholders' domain includes all the software and hardware components that live and operate on the logistics companies' sites. These components can be either owned and operated by the logistics companies or provided and integrated by TRACE. The TRACE platform is the core domain of the system, comprising the backbone that is responsible for integrating data from stakeholders, handling processing and analytics, and optimising operations through intelligent scheduling and route optimisation. Finally, the Open-Source domain offers integration of open-source and third-party data sources into the TRACE platform, enriching its datasets and enhancing its capabilities.

In addition to the horizontal layers, some components have been clustered to formulate logical cross-layer groups. Therefore, the conceptual architecture defines the security, observability, and blockchain groups.

The following sub-sections analyse the components and services per layer and group in the TRACE Cloud platform.

2.3.1 Infrastructure Layer

The primary purpose of this layer is to provide the backbone infrastructure required for deploying, managing, and scaling the TRACE components, ensuring high availability and performance. Some key elements in this layer are:

- **Edge Network / Mesh Network:** Network elements that enable decentralised data processing and communication at the edge (i.e., closer to where the data is generated), reducing latency and bandwidth usage.
- **Edge Nodes, Sensors, and Sensor Managers:** Components and services that are responsible for handling data collection from physical devices (e.g., vehicles, drones, infrastructure sensors).
- **Core Platform:** A Platform-as-a-Service (PaaS) that provides centralised computing and storage capabilities, supporting high-performance data processing and analytics.

- **Computing & Storage:** Elements that ensure adequate computational power and storage capacity for the platform's operations.

2.3.2 Data Management Layer

This layer focuses on handling the flow, storage, and processing of data within the TRACE platform. It ensures efficient data exchange, integration, and management. The key components of this layer are:

- **Cloud-based Data Management System:** A centralised and highly available data lake that supports data storage, querying, and retrieval functionalities. It will be used to store data produced within the context of the TRACE platform, as well as information from external sources.
- **StreamHandler:** A distributed event-driven streaming platform that handles a large amount of heterogeneous data, messages and events, enabling the interconnection of all different data sources in the project. It makes the real-time or near real-time streaming of events from multiple producers to multiple consumers feasible.
- **Interoperability Layer:** A middleware that facilitates seamless data exchange between TRACE components and external systems of the Logistics Companies. It ensures compatibility, data translation, and communication between diverse systems.

2.3.3 Application Layer

This layer hosts the core functional components of TRACE, which focus on enabling the applications and services essential for transportation and logistics optimisation. The key components of this layer are:

- **Scheduler:** The Scheduler is central in coordinating shipment requests and optimising delivery operations. Upon receiving a shipment request from the User Interface, it retrieves the necessary data from the underlying system to compile the input required for optimal scheduling and routing.
- **Route Optimiser:** This component is responsible for calculating the shortest paths with their estimated travel time and distance.
- **Resource and Event Manager:** This component is connected with the StreamHandler to receive sensory and other data and detect any deviations from the transfer plans delivered by the Scheduler and Route Optimiser. It continuously receives input in real-time and makes a decision on whether an event is present, adopting a rule-based approach. Depending on the circumstances, the appropriate mitigation plan may be applied, e.g., trigger the Scheduler for reallocations by

sending the affected shipments and the available vehicles. Finally, it monitors the location of all available vehicles that are active in a specific range of area and can be used in case of reallocations in real-time.

2.3.4 User Interface Layer

This layer provides stakeholders and users with various interfaces to interact with the TRACE platform. The key components of this layer are:

- **API Gateway:** A collection of backend services, acting as a reverse proxy to streamline API calls, aggregate data, and implement common functionalities. These interfaces allow external systems and applications to connect with the TRACE platform, enabling data exchange and integration with third-party services.
- **Lightweight Dashboard:** A lightweight user-friendly interface designed to provide real-time insights and control over various aspects of the TRACE platform. It allows stakeholders to interact with their specific applications and access TRACE functionalities. Additionally, it provides platform administrators with an efficient interface to manage the TRACE system, enabling administrative tasks, configuration, and system monitoring.
- **Virtual Cockpit:** The Virtual Cockpit is a virtual reality application that will offer the following functionalities: i) 3D map route visualisation and current coordinates, ii) Event visualisation through real-time stream connection, iii) Emergency functionality (outbound).

2.3.5 Privacy & Security Group

This cross-layer group protects the integrity, confidentiality, and availability of the TRACE platform's data and services. The main components of this layer are:

- **VPN and API Certificates:** Secure communication channels between TRACE components and external systems, ensuring data is encrypted and authenticated.
- **Data encryption** for the TRACE platform storage: Content encryption for TRACE user data in storage (distributed or centralised), ensuring the data are accessible only to the authorised parties.
- **Intrusion Detection Module**: A classifier acting on top of the streams of sensors/communications to detect potential intrusion and fire mitigation actions. The module will be integrated with the

other components of the system to facilitate the management of recovery activities when intrusion is detected.

- **V2V com security:** A security layer to be integrated into the V2V communication protocol AND a secure protocol allowing for an exchange of keys in an ad-hoc set of nodes belonging to the same platform (e.g., all vehicles of the same logistics company).

2.3.6 Blockchain Group

This group includes components and services utilised for secure data sharing, ensuring immutability and transparency in data exchanges between stakeholders. These elements are:

- **Distributed Ledger Structure:** Mechanisms that allow the TRACE platform to securely connect with the Algorand Blockchain.
- **Digital Wallet:** A system that enables the interface to securely store digital identity credentials.
- **PKI Ecosystem:** A system that facilitates the integration of Public Key Infrastructure (PKI) into existing platforms.
- **Smart Contract:** This component will handle all transactions related to authentication and storage of references on the Blockchain. It serves as the backbone of the system, facilitating seamless communication between all involved components. It is responsible for executing the PKI ecosystem and ensuring the unshakable integrity of collaboration agreements.

2.3.7 Observability and Monitoring Group

This group includes the following components that allow administrators to monitor, analyse, and maintain the health of the TRACE platform:

- **Analytics:** Services that Provide insights into system performance, user behaviour, and operational efficiency.
- **Application & Storage Monitoring:** It continuously tracks the performance of applications and storage systems, helping to identify and resolve issues proactively.
- **Infrastructure Monitoring:** A core service that interacts with the infrastructure layer elements to ensure that the underlying hardware and network infrastructure are functioning optimally.

2.4 TRACE Detailed Architecture Diagram

The detailed TRACE Architecture Diagram extends the high-level conceptual architecture by providing a concrete and clear visual representation of the platform's structure and deployment view. The diagram (presented in Figure 6) illustrates the interaction between the platform components deployed in its key domains: the Stakeholders Domain, the Cloud Platform, and the Public Ledger. Each domain contributes distinct functionalities, with the Stakeholders Domain focusing on vehicle and infrastructure integration, the Cloud Platform managing data processing and decision-making, and the Public Ledger ensuring secure and transparent transactions through blockchain technology.

This detailed architecture diagram offers a detailed map of the platform's components, their internal modules, and their integration points. The purpose of this diagram is to provide a clear and comprehensive overview of the TRACE platform's operational framework, guiding the development, integration, and deployment of its building blocks while ensuring alignment with the project's objectives. More details about the components, their interfaces, internal modules, and technologies can be found in the deliverables mentioned in the respective tables in the next Section.

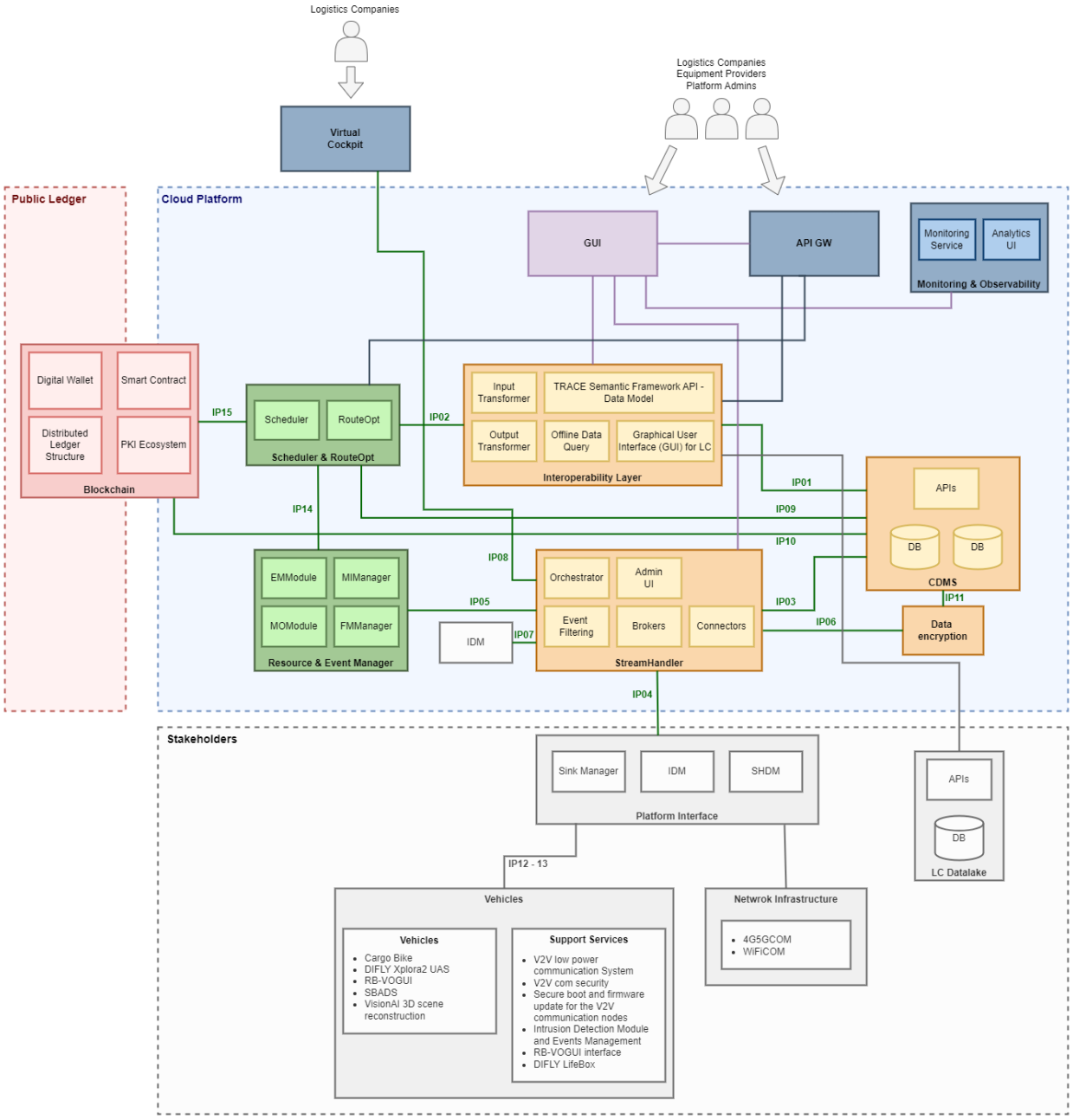

Figure 6: TRACE Detailed Architecture Diagram

2.5 TRACE Platform Interfaces

Following the platform dataflows and architecture definition, all the integration points amongst the various TRACE components were identified and documented. For this purpose, the technical teams created an integration matrix based on the format typically used in a Design Structure Matrix (DSM). A DSM is a technique commonly used in systems integration to model the structure of complex systems and the related interactions. A design structure matrix is a square matrix representing linkages between the system elements. The system elements are often labelled in the rows to the left of the matrix and/or in the columns above the matrix. These elements represent individual stand-alone components of the TRACE platform.

Table 1 lists all constituent TRACE components and specifies the corresponding interfaces amongst them:

Table 1: Integration Matrix

Component	Int Layer	SH	CDMS	V2V LPCS	Platform Iface	Vis 3D SR	REManager	Sch & RouteOpt	Blockchain	Data Enc	IDM	Virt Cockpit
Interoperability Layer			IP01					IP02				
StreamHandler			IP03		IP04		IP05			IP06	IP07	IP08
CDMS	IP01	IP03						IP09	IP10	IP11		
V2V low power com sys					IP12							
Platform Interface		IP04		IP12		IP13						
Vision 3D scene registration					IP13							
Resource & Event Manager		IP05						IP14				
Scheduler & Route Opt.	IP02		IP09				IP14		IP15			
Blockchain			IP10					IP15				
Data Encryption		IP06	IP11									
IDM		IP07										
Virtual Cockpit		IP08										

The following subsection describes all the identified integration points as documented by the responsible technical teams. The template Table 2 has been created to capture the relative information for each integration point:

Table 2: Integration Point Documentation Template

Identifier	*This field provides a discrete identifier to each integration point exclusively and eases its reference for purposes internal to the project. The identifiers follow the convention IP[NN]_[COMP1]-[COMP2].*
Responsible	*Partners that are responsible for documenting, implementing, and testing the integration point.*
Integration Point Purpose	*A summary explaining the exact interaction and the information exchanged between the components, as well as the purpose that it serves.*
Interface Type	*The technology used for the implementation of the interface, e.g., Pub/Sub, REST, Serial, etc.*
References	*A list of references to existing TRACE deliverables that contain information on the specific interaction, such as interface specifications.*
Notes	*Additional information or notes included by the partner leading the components' development.*

2.5.1 Integration Points

Table 3: Interoperability Layer - - Cloud-based Data Management System > IP01 integration point

Identifier	IP01_IntLayer-CDMS
Responsible	INTRA, NKUA
Integration Point Purpose	Harmonise data from the various logistics companies that will be permanently stored in the TRACE storage system.
Interface Type	REST
References	D3.1

Notes	It refers to the data that is imported to the TRACE storage through the UI.

Table 4: Interoperability Layer - - Scheduler & Route Opt. > IP02 integration point

Identifier	IP02_IntLayer-Sch
Responsible	NKUA, TUC
Integration Point Purpose	A connection with the Interoperability Layer provides the Scheduler with input data to calculate the optimal output.
Interface Type	Pub/Sub
References	D3.1, D4.3
Notes	All the real-time data will be provided through the connectors (Southbound) of the logistics companies.

Table 5: - Scheduler & Route Opt. - - Cloud-based Data Management System > IP03 integration point

Identifier	IP03_SH-CDMS
Responsible	INTRA
Integration Point Purpose	Permanently store data that goes through the various StreamHandler topics in the TRACE storage system.
Interface Type	Pub/Sub
References	D3.1
Notes	

Table 6: StreamHandler - - Platform Interface > IP04 integration point

Identifier	IP04_SH-PlatformIface
Responsible	INTRA, NKUA
Integration Point Purpose	Receive streams of data from the various hardware and software components of the infrastructure layer, e.g., sensors, UAVs, etc.
Interface Type	Pub/Sub
References	D3.1, D4.1
Notes	The Platform Interface acts as a data sink that integrates data streams from various sources of the infrastructure layer and sends them to the StreamHandler.

Table 7: StreamHandler - - Resource & Event Manager > IP05 integration point

Identifier	IP05_SH-REManager
Responsible	INTRA, UTH
Integration Point Purpose	The StreamHandler sends the various streams of data to the Resource and Event Manager, which is responsible for dispatching the messages to the required components of the platform.
Interface Type	Pub/Sub
References	D3.1, D4.1
Notes	

Table 8: StreamHandler - - Data Encryption > IP06 integration point

Identifier	IP06_SH-DataEnc
Responsible	INTRA, SID

Integration Point Purpose	The Data Encryption component will receive messages from the SH that will encrypt using encryption keys. Consumers must use their own keys to decrypt the messages.
Interface Type	Pub/Sub
References	D3.1, D4.5
Notes	

Table 9: StreamHandler - - IDM > IP07 integration point

Identifier	IP07_SH-IDM
Responsible	INTRA, UNIS
Integration Point Purpose	The IDM receives real-time sensor data streams to classify as normal or abnormal, detecting intrusions based on patterns or anomalies.
Interface Type	Pub/Sub
References	D3.1, D4.5
Notes	

Table 10: StreamHandler - - Virtual Cockpit > IP08 integration point

Identifier	IP08_SH-VirtCockpit
Responsible	INTRA, CDW
Integration Point Purpose	The Virtual Cockpit receives streams of data coming from vehicles and sensors of the infrastructure layer. These data are part of the vehicles' control plane (such as GPS signal) and multimedia data.
Interface Type	Pub/Sub
References	D3.1, D4.5

Notes	

Table 11: Cloud-based Data Management System - - Scheduler & Route Opt. > IP09 integration point

Identifier	IP09_CDMS-Sch
Responsible	INTRA, TUC
Integration Point Purpose	The Scheduler and Route Optimiser components send queries to the TRACE storage system in order to retrieve stored data related to the available resources, such as available UAVs. Moreover, these components use the CDMS to store shipment-related data permanently.
Interface Type	REST
References	D3.1, D4.3
Notes	

Table 12: Cloud-based Data Management System - - Blockchain > IP10 integration point

Identifier	IP10_CDMS-Blockchain
Responsible	INTRA, BC5
Integration Point Purpose	The Blockchain component interacts with the CDMS to store data associated with the generated dNFT of the shipments.
Interface Type	REST
References	D3.1, D4.3
Notes	

Table 13: Cloud-based Data Management System - - Data Encryption > IP11 integration point

Identifier	IP11_CDMS-DataEnc

Responsible	INTRA, SID
Integration Point Purpose	The Data Encryption component is used to encrypt data at rest, stored in the CDMS.
Interface Type	REST
References	D3.1, D4.5
Notes	Part of the Data Encryption functionality will be integrated into the API GW module of the CDMS.

Table 14: V2V low power com sys - - Platform Interface > IP12 integration point

Identifier	IP12_V2VLPCS-PlatformIface
Responsible	CSEM, UNIMORE
Integration Point Purpose	The Vehicle-to-Vehicle components is used to send relevant data between the cargo bike composing the platoon. It needs to be interfaced with the device able to generate and treat this data on each cargo bike.
Interface Type	Serial
References	D4.1
Notes	This communication link will also be used to update the software on the V2V communication module.

Table 15: Vision 3D scene registration - - Platform Interface > IP13 integration point

Identifier	IP13_Vis3DSR-PlatformIface
Responsible	CSEM, UNIMORE
Integration Point Purpose	The 3D scene reconstruction module is intended to provide a better understanding of the scenes for the platoon bikes in real time, allowing for improved obstacle detection and avoidance. It needs to interface with the Bike Connection Box (developed by

	UNIMORE) which manages transfer of information between different vehicles in a platoon formation.
Interface Type	Pub/Sub
References	D4.1
Notes	The system uses edge processing to handle data locally, reducing latency and safeguarding sensitive information by avoiding transmission to centralized servers.

Table 16: Scheduler & Route Opt. - - Resource & Event Manager > IP14 integration point

Identifier	IP14_Sch-REManager
Responsible	TUC, UTH
Integration Point Purpose	When a delivery is disrupted, the REManager requests the Scheduler to change course or assign optimal routes to vehicles that pick up sensitive shipments if the current mode of transportation has broken down.
Interface Type	REST
References	D4.3
Notes	

Table 17: Scheduler & Route Opt. - - Blockchain > IP15 integration point

Identifier	IP15_Sch-Blockchain
Responsible	TUC, BC5
Integration Point Purpose	The Scheduler interacts with the blockchain to generate new delivery requests and manage the secure storage of schedules. This includes both proposed schedules generated by the platform and approved schedules confirmed by the user.
Interface Type	REST

References	D4.3
Notes	

3 Interoperability Layer

The TRACE Interoperability Layer addresses the critical challenge of integrating data from disparate operational data sources in the logistics ecosystem. Each data source that is maintained by different logistics companies operates in isolation and "speaks" its own language. These databases vary significantly in terms of underlying technology, such as relational databases (RDBMS) versus NoSQL systems, and use different frameworks like MySQL or PostgreSQL. Additionally, they follow distinct schemas, further complicating integration efforts, as similar concepts are often described using inconsistent terminologies. For instance, some databases may use the term "vehicles" while others may use the term "trucks" to represent the same entity. This lack of standardization creates significant barriers to collaboration and data sharing, as systems cannot inherently understand or interact with each other. Without a unified approach, every query would require manual adaptation for each individual database, leading to inefficiencies, data silos, and limited scalability.

The proposed solution is the TRACE Data Interoperability Layer, which acts as an intermediary to integrate and unify these diverse data models. All TRACE components communicate using a common TRACE Language which is based on the TRACE Data Model. When a module in TRACE submits a query, such as requesting data about available vehicles for an optimization process, the interoperability layer transforms the query into a set of database-specific queries that can be executed on each operational database. The results from these data sources are then collected, harmonized to resolve differences in schema and terminology, and returned as a cohesive dataset.

3.1 Conceptual Design

The TRACE Interoperability layer addresses the growing need for seamless collaboration and integration in the logistics industry, particularly in the context of multi-modal transportation of goods and resource optimization. As logistics companies increasingly face challenges such as managing diverse transportation modes (e.g., unmanned vehicles, manned vehicles, cargo bikes, bicycles), coordinating shipments across multiple stakeholders, and optimizing delivery times while minimizing costs, the TRACE Interoperability layer offers a robust solution. Central to this layer is the TRACE data model, which provides a structured and unified framework to capture, organize, and analyze critical logistics data.

The two figures below illustrate the semantic integration process in TRACE, designed to enable collaboration and data querying across diverse logistics data sources. The process is divided into two interconnected phases: the Preparatory Phase which establishes the foundational semantic alignment between the TRACE data model and the models of the underlying sources and the Query Phase where users retrieve harmonized data through the TRACE platform.

The Preparatory Phase (Figure 7) focuses on integrating the heterogeneous data models of logistics databases with the TRACE Data Model. The TRACE model serves as the semantic backbone of the system, providing a unified vocabulary and structure for describing and querying logistics data. This phase begins with the retrieval of dataset schemas from the logistics databases, which are provided by the companies. These schemas are transformed into semantic models through a process called semantic annotation, which maps the data elements to the concepts defined in the TRACE model. This step ensures that the diverse datasets align with the TRACE ontology while avoiding data replication (i.e., there is no need to import data into TRACE system).

Each model is linked to the TRACE model through a software connector. For every mapping, essential metadata, such as units of measurement or contextual definitions, is generated and stored to maintain consistency and accuracy. These connectors translate queries and results between the TRACE model and the specific database schemas, effectively bridging the gap between the heterogeneous datasets and the common language of TRACE. Once this integration is complete, the TRACE model is exposed through an endpoint, enabling users and systems to interact with the platform.

The Query Phase (Figure 8) builds upon the groundwork established in the preparatory phase, allowing users to query the integrated logistics data sources. Users, who may be non-technical far-end users or programmers (i.e., software). Their queries are submitted to a centralized query endpoint, which serves as the main access point to the system. The query is processed and passed to the TRACE Interoperability Layer which translates it into data source-specific terms using the connectors layer.

The data model then constructs queries for each relevant dataset, retrieving raw data from the logistics databases. Finally, the harmonized data is merged into a unified format and returned to the user through the query endpoint, providing integrated results that are easy to interpret and act upon.

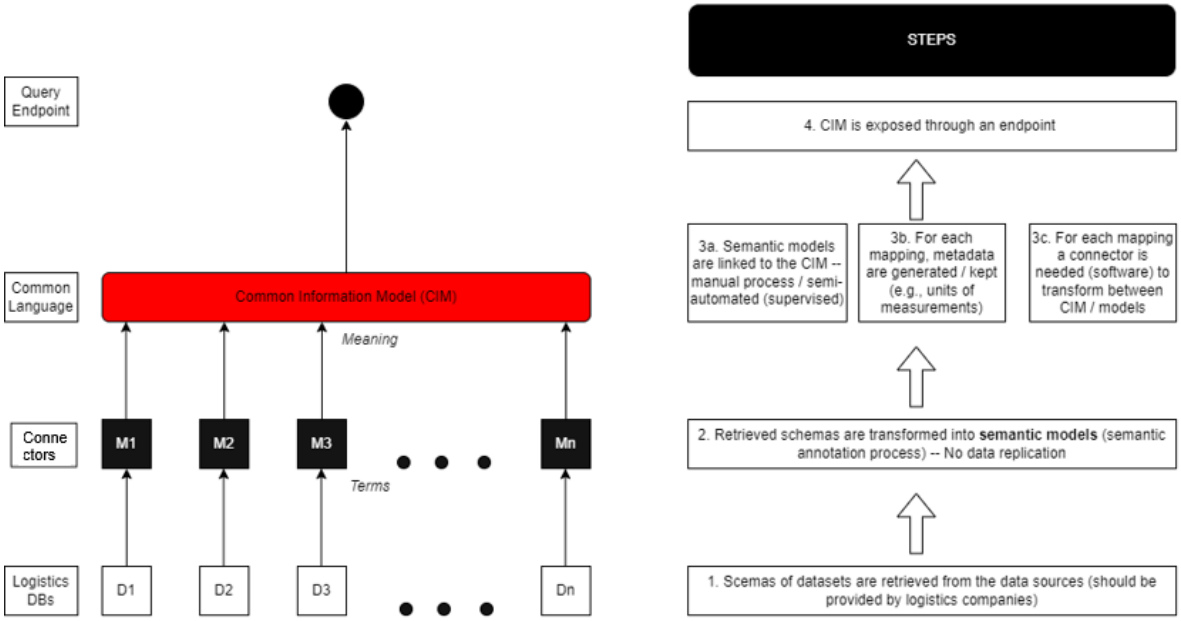

Figure 7: TRACE Semantic Integration approach – Integration of Logistics DBs

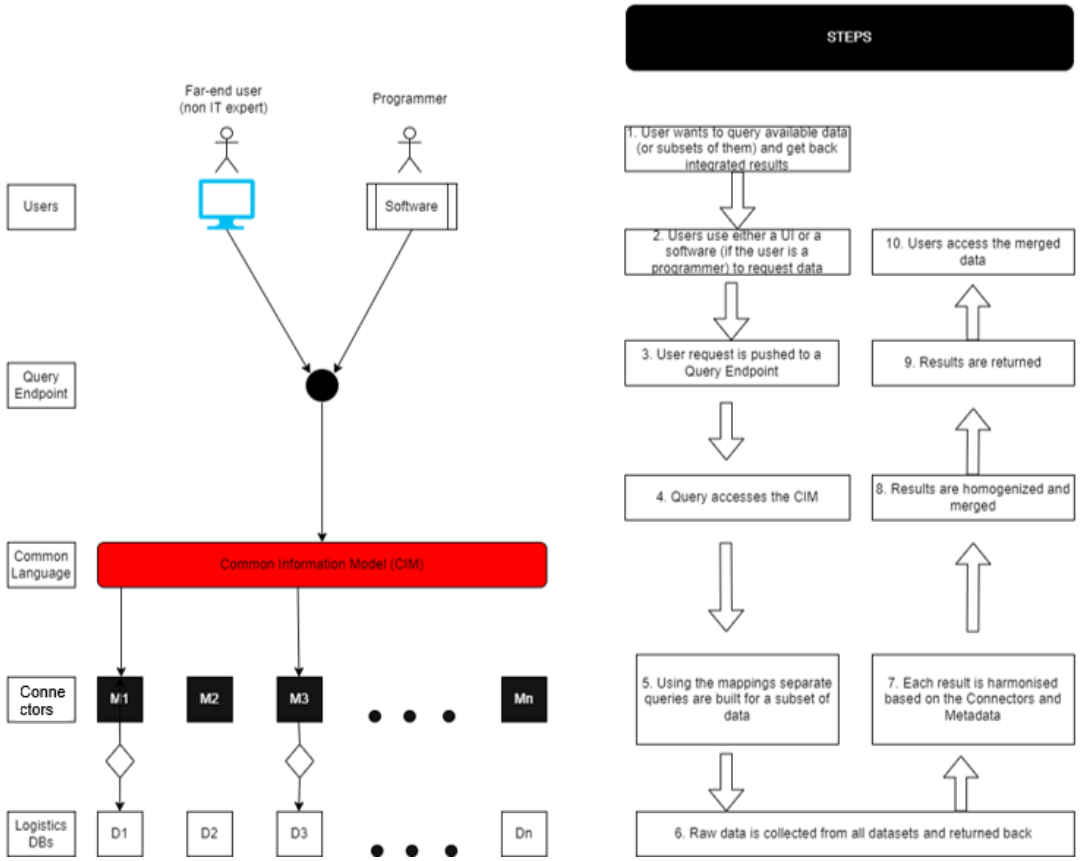

Figure 8: TRACE Semantic Integration approach – Query Logistics DBs

3.2 TRACE Data Model

Figure 9 visualizes the core data classes and relationships TRACE analytical data model.

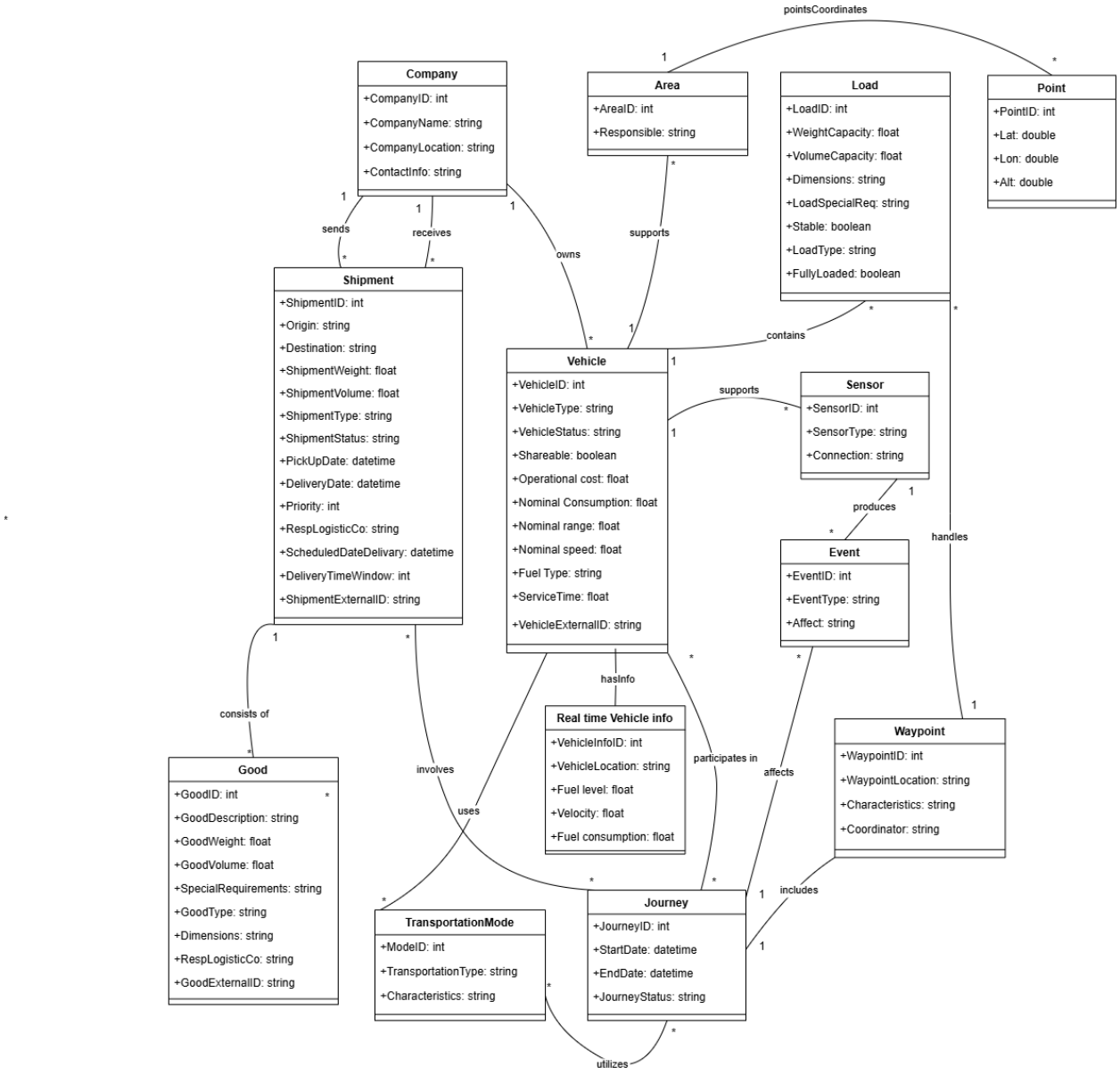

Figure 9: TRACE analytical data model

Here is the detailed description of the analytical data model for the TRACE platform, designed to enable collaboration between logistics companies for multi-modal transportation and delivery of goods. The model integrates data from shipments, resources, vehicles, and companies, and its structure is expressed in UML format. In the tables that follow, there is a short description of the main classes, attributes and relationships.

Table 18: Attributes of Class 'Company'

Attribute	Data Type	Unit	Description
CompanyID	int	-	Unique identifier for the company, created by TRACE.
CompanyName	string	-	Name of the company.
CompanyLocation	string	WGS84	Geographic location of the company as WGS84 coordinates.
ContactInfo	string	-	Contact information of the company.

Table 19: Attributes of Class 'Area'

Attribute	Data Type	Unit	Description
AreaID	int	-	Unique identifier for the area, created by TRACE.
Responsible	string	-	Person or entity responsible for this area.

Table 20: Attributes of Class 'Point'

Attribute	Data Type	Unit	Description
PointID	int	-	Unique identifier for the point, created by TRACE.
Lat	double	degrees	Latitude of the point in decimal degrees.
Lon	double	degrees	Longitude of the point in decimal degrees.
Alt	double	meters	Altitude of the point in meters above sea level.

Table 21: Attributes of Class 'Shipment'

Attribute	Data Type	Unit	Description
ShipmentID	int	-	Unique identifier for the shipment, created by TRACE.
Origin	string	-	Address of the shipment's origin.
Destination	string	-	Address of the shipment's destination.
ShipmentWeight	float	kg	Total weight of the shipment.
ShipmentVolume	float	cm^3	Total volume of the shipment.
ShipmentType	string	-	Classification of the shipment (e.g., standard, fragile).

Attribute	Data Type	Unit	Description
ShipmentStatus	string	-	Current status of the shipment (e.g., ongoing, delivered).
PickupDate	datetime	CET	Date and time the shipment is picked up.
DeliveryDate	datetime	CET	Date and time the shipment is delivered.
Priority	int	-	Priority level of the shipment (1: urgent, 5: normal).
RespLogisticCo	string	-	Responsible logistics company.
ScheduledDateDelivery	datetime	CET	Scheduled delivery date and time.
DeliveryTimeWindow	int	minutes	Time window for the delivery.
ShipmentExternalID	string	-	External reference for the shipment.

Table 22: Attributes of Class 'Good'

Attribute	Data Type	Unit	Description
GoodID	int	-	Unique identifier for the good, created by TRACE.
GoodDescription	string	-	Short description of the good.
GoodWeight	float	kg	Weight of the good.
GoodVolume	float	cm^3	Volume of the good.
SpecialRequirements	string	-	Any special requirements for handling the good.
Dimensions	string	cm	Dimensions of the good (length, width, height).
GoodType	string	-	Classification of the good (e.g., standard, fragile).
RespLogisticCo	string	-	Responsible logistics company for the good.
GoodExternalID	string	-	External reference for the good.

Table 23: Attributes of Class 'Vehicle'

Attribute	Data Type	Unit	Description
VehicleID	int	-	Unique identifier for the vehicle, created by TRACE.
VehicleType	string	-	Type of vehicle (e.g., UAV, truck, bicycle).
VehicleStatus	string	-	Current status of the vehicle (e.g., operating, damaged).

Attribute	Data Type	Unit	Description
Shareable	boolean	-	Indicates if the vehicle is shareable.
OperationalCost	float	euro/hour	Cost of operating the vehicle.
NominalConsumption	float	fuel/hour	Fuel or energy consumption rate.
NominalRange	float	m	Range the vehicle can travel on a full tank/battery.
NominalSpeed	float	m/s	Average operational speed of the vehicle.
FuelType	string	-	Type of fuel used (e.g., gasoline, battery).
ServiceTime	float	seconds	Average service time for maintenance or recharging.
VehicleExternalID	string	-	External reference for the vehicle.

Table 24: Attributes of Class 'Journey'

Attribute	Data Type	Unit	Description
JourneyID	int	-	Unique identifier for the journey, created by TRACE.
StartDate	datetime	CET	Date and time the journey begins.
EndDate	datetime	CET	Date and time the journey ends.
JourneyStatus	string	-	Current status of the journey (e.g., active, completed).

Table 25: Attributes of Class 'Load'

Attribute	Data Type	Unit	Description
LoadID	int	-	Unique identifier for the load, created by TRACE.
WeightCapacity	float	kg	Maximum weight capacity of the load.
VolumeCapacity	float	cm^3	Maximum volume capacity of the load.
Dimensions	string	cm	Dimensions of the load.
Stable	boolean	-	Indicates if the load is stable.
LoadType	string	-	Type of load being carried.
FullyLoaded	boolean	-	Indicates if the vehicle is fully loaded.

Table 26: Attributes of Class 'Sensor'

Attribute	Data Type	Unit	Description
SensorID	int	-	Unique identifier for the sensor, created by TRACE.
SensorType	string	-	Type of sensor (e.g., GPS, temperature, vibration).
Connection	string	-	Connection type used by the sensor (e.g., wireless, wired).

Table 27: Attributes of Class 'Event'

Attribute	Data Type	Unit	Description
EventID	int	-	Unique identifier for the event, created by TRACE.
EventType	string	-	Type of event (e.g., breakdown, delay, temperature threshold exceeded).
Affect	string	-	Description of how the event impacts logistics operations.

Table 28: Attributes of Class 'Waypoint'

Attribute	Data Type	Unit	Description
WaypointID	int	-	Unique identifier for the waypoint, created by TRACE.
WaypointLocation	string	WGS84	Location of the waypoint using WGS84 coordinates.
Characteristics	string	-	Additional descriptive details about the waypoint.
Coordinator	string	-	Person or system managing the waypoint.

Table 29: Attributes of Class 'Transportation Mode'

Attribute	Data Type	Unit	Description
ModeID	int	-	Unique identifier for the transportation mode, created by TRACE.
TransportationType	string	-	Type of transportation (e.g., road, rail, air, sea).
Characteristics	string	-	Specific features or details about the mode of transportation.

Table 30: Attributes of Class 'Real-Time Vehicle Info'

Attribute	Data Type	Unit	Description
VehicleID	int	-	Identifier of the vehicle for which real-time data is being logged.
VehicleLocation	string	WGS84	Current location of the vehicle using WGS84 coordinates.
FuelLevel	float	%	Current fuel or energy level of the vehicle, as a percentage.
Velocity	float	m/s	Current speed of the vehicle.
FuelConsumption	float	fuel/hour	Current rate of fuel or energy consumption.

Relationships Between Classes:

1. Company ↔ Shipment
   - Sends: A company sends shipments (1:N relationship).
   - Receives: A company receives shipments (1:N relationship).
2. Company ↔ Vehicle
   - Owns: A company owns vehicles (1:N relationship).
3. Area ↔ Point
   - pointsCoordinates: An area is defined by multiple points (1:N relationship).
4. Area ↔ Vehicle
   - Supports: An area is supported by multiple vehicles (1:N relationship).
5. Shipment ↔ Good
   - Consists Of: A shipment consists of one or more goods (1:N relationship).
6. Shipment ↔ Journey
   - Involves: A shipment involves one or more journeys (N:M relationship).
7. Vehicle ↔ Sensor
   - Supports: A vehicle is equipped with sensors (1:N relationship).
8. Sensor ↔ Event
   - Produces: A sensor produces events (1:N relationship).
9. Load ↔ Waypoint
   - Handles: Each load is associated with zero or more waypoints (N:1 relationship).
10. Journey ↔ Waypoint
    - Includes: A journey includes waypoints (1:N relationship).
11. Journey ↔ Transportation Mode
    - Utilizes: A journey utilizes one or more transportation modes (1:N relationship).
12. Journey ↔ Vehicle
    - participatesIn: A vehicle participates in one or more journeys (N:M relationship).
13. Vehicle ↔ Real-Time Vehicle Info
    - Has Info: A vehicle has associated real-time information (1:1 relationship).

14. Load ↔ Vehicle
    - Contains: A load is carried by a vehicle (N:1 relationship).
15. Vehicle ↔ Transportation Mode
    - Supports: A vehicle uses one or more transportation modes (1:N relationship).
16. Event ↔ Journey
    - Affects: Each journey is affected by multiple events (N:1 relationship).

3.3 TRACE Ontology

The TRACE Interoperability layer is powered by a semantic framework, the TRACE ontology, which acts as a common language for all TRACE modules to interact in a uniform and standardized way with underlying data sources from logistics companies. By leveraging semantic technologies, the TRACE ontology ensures that data related to shipments, resources, vehicles, and journeys is described in a consistent and machine-readable manner, enabling seamless communication between heterogeneous systems. This approach mitigates interoperability issues arising from differences in data structures, formats, and terminologies across various stakeholders and data providers.

The TRACE ontology also facilitates dynamic data integration by mapping diverse data sources into a cohesive framework, enabling the TRACE platform to support real-time operations, predictive analytics, and decision-making. It acts as the backbone of the platform, allowing modules to query and share information efficiently while maintaining data accuracy and consistency. Through this integration, logistics companies can achieve efficient resource sharing, improved route planning, enhanced real-time monitoring, and informed decision-making, ultimately fostering greater efficiency, transparency, and collaboration within the logistics ecosystem.

Figure 10 shows the main hierarchy of the TRACE ontology while Figure 11 visualizes its basic attributes and relationships between main classes using the VOWL plugin [3].

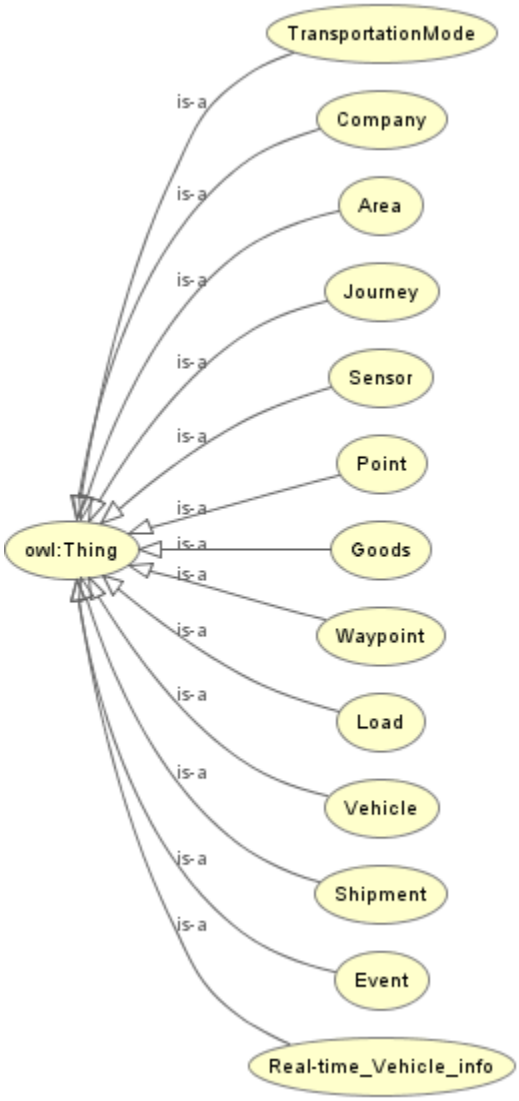

Figure 10: TRACE ontology basic class hierarchy

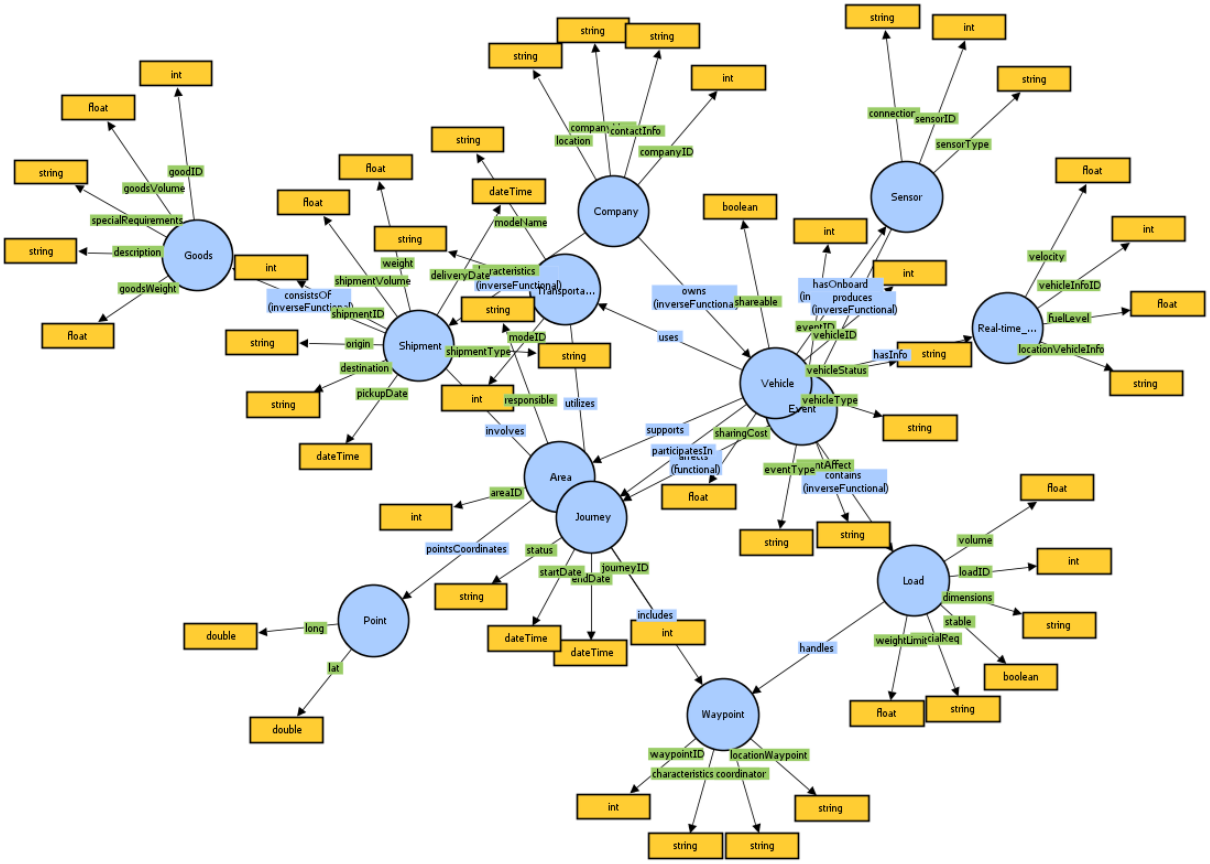

Figure 11: TRACE ontology basic attributes and relationships visualization

The tables below show the statistics of the TRACE ontology.

Table 31: Statistics of the TRACE ontology

Metric	Count
Axioms	224
Logical Axiom Count	150
Declaration Axioms Count	68
Class Count	13
Object Property Count	17
Data Property Count	54

Table 32: Object Property Axioms

Axiom Type	Count
ObjectPropertyDomain	17
ObjectPropertyRange	17
FunctionalObjectProperty	1
InverseFunctionalObjectProperty	7

Table 33: Data Property Axioms

Axiom Type	Count
DataPropertyDomain	54
DataPropertyRange	54

3.4 Implementation View

This section describes the practical implementation of the TRACE Interoperability Layer within the broader system architecture of the TRACE project. Figure 12 demonstrates how the conceptual framework of data interoperability is operationalized to integrate heterogeneous data sources and facilitate seamless communication between TRACE components. At the core of this architecture is the TRACE Interoperability Layer (developed by NKUA), which acts as the central mediator for integrating data from multiple, heterogeneous sources into the unified TRACE Data Model. This layer exposes REST APIs to interact with other TRACE components, such as the User Interface (UI) developed by UTH, the Scheduler developed by TUC, and a Blockchain module by BC5. The REST APIs enable these components to communicate seamlessly with the Interoperability Layer, which manages the data integration and standardization processes. The TRACE Storage (managed by Netcompany) serves as the centralized repository where harmonized and transformed data is stored. This repository interacts with the Interoperability Layer to fetch or update data as required. Beneath the Interoperability Layer, the architecture incorporates multiple connectors tailored to specific data providers:

1. ACS-CERTH Connector:
    a. Retrieves data from ACS's operational systems via Excel files or APIs.

b. Transforms the data into the TRACE Data Model format through the connector's transformation logic.

2. HT-UTH Connector:
   a. Similar to the CERTH connector, it processes Excel data provided by UTH.
   b. Performs schema transformations to align the data with the TRACE Data Model.
3. Slovenian Post - UGLA Connector:
   a. Interacts with Slovenian Post's systems using Excel files or APIs.
   b. Applies transformation logic to ensure compatibility with the TRACE Data Model.

In addition to these core data providers, the architecture includes integration with third-party systems such as URBICO, SUM SOLUTIONS, UNIMORE, ROBOTNIK, and DIFLY. These systems interact with the TRACE platform through the Blockchain module and the Interoperability Layer via their respective REST APIs, ensuring secure and efficient data sharing.

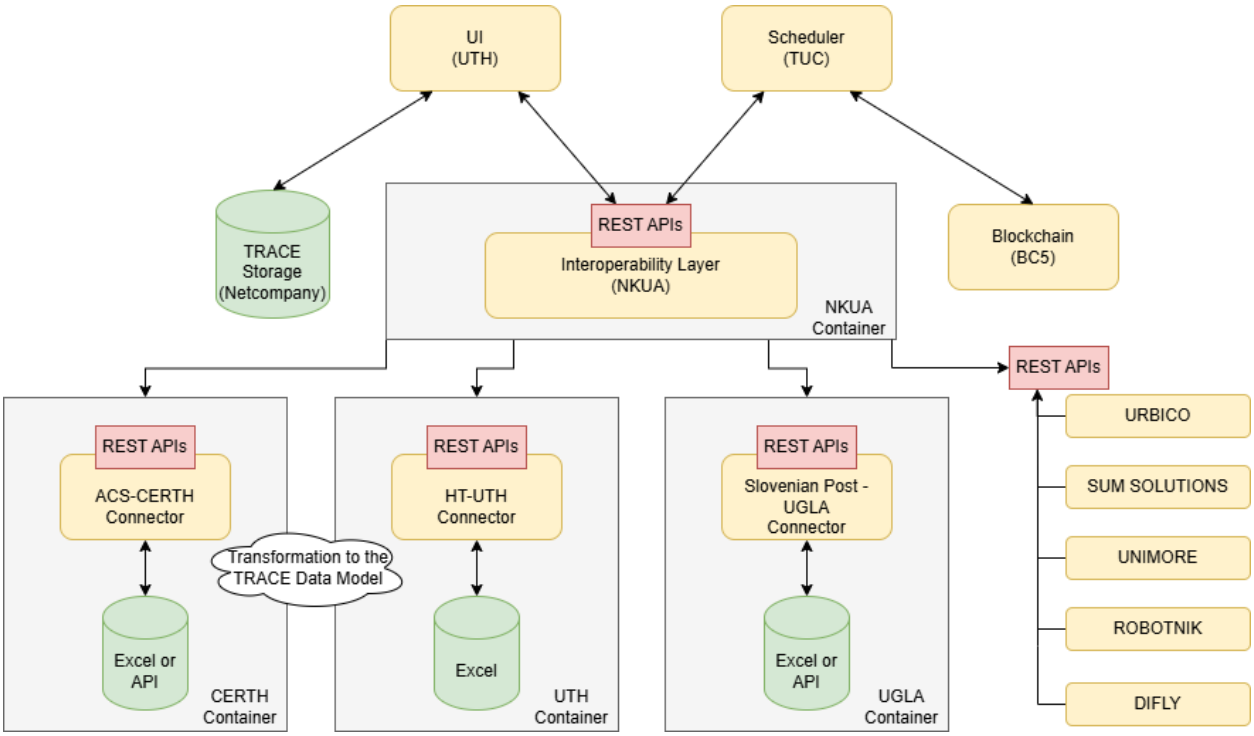

Figure 12: TRACE Interoperability Layer – Implementation View

3.5 Integration Pathways with External Traffic and Weather Management Systems

To further extend TRACE's interoperability capabilities, specific integration pathways have been defined to ensure compatibility with existing Traffic Management Systems (TMS), Weather Management Systems (WMS), and middleware solutions used by authorities and logistics stakeholders. These external data sources can significantly enhance the performance of the TRACE Route Optimiser, Scheduler, and Fleet Monitoring components by introducing real-time, context-aware parameters into the decision-making process.

By developing an API capable of receiving data from platforms such as Google Maps or Traffic Management Systems (TMS) operated by local authorities, the solution can enhance route optimisation for vehicles. These enhancements include accurate estimated travel times, as it is calculated on real-time data, and avoiding congestion. Furthermore, data from Weather Management Systems can be incorporated into the framework to adjust vehicle speeds and improve the accuracy of delivery time predictions. For ground vehicles, this adjustment can be made based on weather-induced speed variations, while for aerial vehicles and unmanned ground vehicles, environmental factors such as wind, precipitation, fog, visibility, temperature extremes, and humidity can be accounted for by evaluating logic functions for the available types of aerial vehicles and their capabilities.

To support these capabilities, it is recommended to connect the Route Optimizer's ground vehicle module with the TMS API, allowing the direct import of real-time traffic information into the route calculation process. In addition, a dedicated component should be developed to evaluate environmental conditions. This component would interface with both the Fleet Monitoring Module, which is responsible for the vehicle selection and the Route Optimizer, enabling dynamic decision-making.

From a technical integration perspective, the Interoperability Layer can connect with these external systems through standardised API protocols and middleware connectors, such as REST, MQTT, and NGSI-LD. These connectors will allow TRACE to both consume and publish relevant operational data, ensuring bidirectional data exchange with external platforms, including city-level TMS, meteorological networks, and public information services.

To guarantee semantic alignment, the retrieved data could be linked to the TRACE Data Model, maintaining consistency with internal representations used by the Scheduler, Route Optimiser, and Big Data

Management subsystems. This way, the system ensures that external context data such as traffic congestion levels or severe weather alerts can be seamlessly fused with operational data already collected within TRACE, thereby enabling real-time adaptive logistics planning. Through this approach, the Interoperability Layer can serve as the central gateway for integrating external environmental and infrastructure information into the TRACE ecosystem, strengthening the platform's adaptability, resilience, and predictive decision-support capabilities.

4 Big Data Management System

The TRACE platform is equipped with a big data management system, which is responsible for integrating and handling large amounts of heterogeneous data coming from various sources. To achieve this, the big data management system utilises two components that extend the interoperability layer presented in the previous section: The StreamHandler and the Cloud-based Data Management System.

4.1 StreamHandler

The StreamHandler is a critical component of the TRACE platform, designed to handle large-scale, heterogeneous data streams in real time. It acts as a distributed event-driven streaming platform that facilitates the collection, integration, and processing of data from various sources, such as IoT devices, sensors, vehicles, and backend systems. By enabling seamless communication and ensuring standardised interfaces for data exchange, the StreamHandler plays a critical role in maintaining interoperability across the platform.

Through connectors and producers, it gathers data streams, making them available to other components of the TRACE platform for further analysis and action. This is crucial for real-time logistics operations, such as monitoring vehicle status, tracking shipments, and responding to traffic disruptions. The StreamHandler supports a publish-subscribe communication pattern, enabling producers to send data streams to which consumers can subscribe, ensuring reliable, ordered delivery of events.

In the context of TRACE, the StreamHandler platform is used to interconnect diverse vehicle sensors and logistics companies' systems deployed at their premises with the centralised cloud platform for data exchange purposes. The exchanged data will adhere to the defined data model, as presented in Section 3.2. The following elements/classes of the data model will be processed through the Streamhandler: Event, Real-Time Vehicle Info, and Journey. In addition to that, it acts as an integration point for other internal platform components that interact with each other using a Pub/Sub communication model.

The internal architecture of the StreamHandler is illustrated in the Figure below:

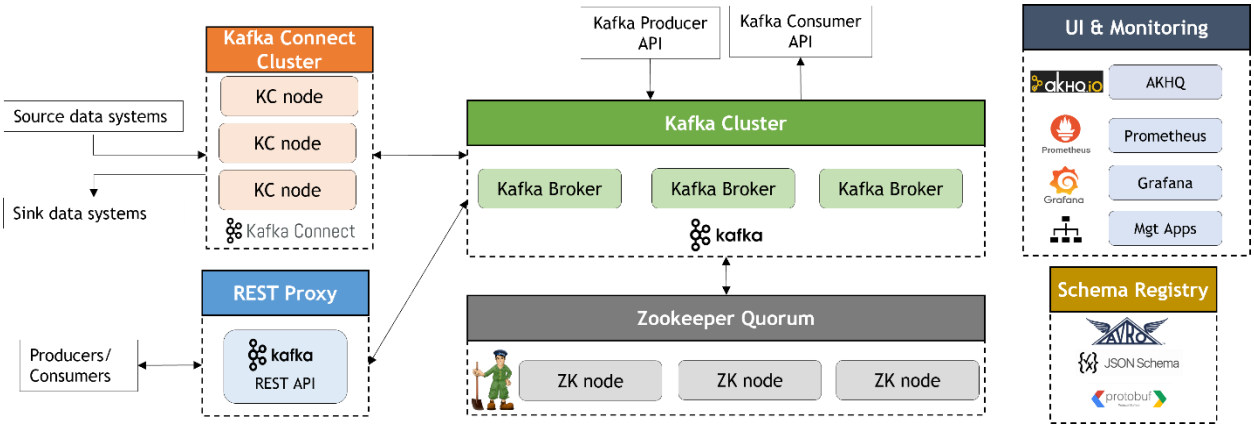

Figure 13: StreamHandler Architecture

The following subsections present the various internal modules of the StreamHandler.

4.1.1 Client API

Other platform components are connected to the StreamHandler via client APIs. The most popular APIs are Kafka Producers, Kafka Consumers, and Kafka Connect. Producers ingest a stream of messages to one or more Kafka topics. Producer API enables developers to create custom producers that write data to Kafka topics. The API provides several options for configuring the behaviour of the producer, such as data serialisation, data compression, etc. On the other hand, consumers subscribe to one or more topics to automatically receive streams of ingested messages. The Consumer API enables developers to create their own consumers to read data from Kafka topics. It provides several options for configuring the behaviour of the consumer, such as setting the position to start consuming from or setting the number of records to fetch at a time.

4.1.2 Kafka Cluster

The Kafka Cluster [4] is the heart of the StreamHandler. It is comprised of multiple brokers that work together to distribute load and ensure reliable redundancy and high availability. The brokers are responsible for storing the incoming data and allowing both producers and consumers to connect to the system. Kafka stores this data in streams of messages or records that belong to a particular type, which is called a Topic. Each message contains a key, the actual value, and a timestamp. The data provider defines the key and the value fields of the message, while Kafka adds the timestamp upon receiving the message. Apache Zookeeper is used to coordinate the deployed brokers and manage the cluster quorum. Each broker has a unique ID and is responsible for storing partitions of one or more topics, while a leader broker

is assigned to each topic. This ensures that the system will continue operating without any issues or data loss, even if a worker goes down.

4.1.3 Zookeeper

The Kafka cluster and its brokers are managed and coordinated by Apache Zookeeper [5]. ZooKeeper is an open-source distributed coordination service designed to manage and synchronise distributed systems. It monitors the system and detects changes to the brokers' topology or topics. For example, Zookeeper detects when a new broker joins the cluster, or when a broker experiences a failure. It then notifies all the system nodes about the updates and the new system topology. Moreover, it enables leadership elections amongst brokers and topic partition pairs. Through this process, the system decides which brokers hold replicas of the same data and which broker will be in charge of a given partition, as well as read and write operations from producers and consumers. Upon a system update, the brokers are notified by the Zookeeper to coordinate with each other and elect any new partition leaders wherever required. This ensures that the Kafka cluster remains operational, safeguarding against a broker's unexpected absence.

4.1.4 Kafka Connect

Kafka Connect [6] is another core module of the StreamHandler that is designed to simplify and automate the process of streaming data between Kafka and other systems, such as databases, file systems, message queues, and other data storage systems. It acts as a middleware, enabling efficient data ingestion and export without the need to write custom code for each integration. To achieve that, it offers a set of tools, known as connectors, that define how data is read from or written to an external system. Each connector is divided into tasks that perform the actual work of pulling or pushing data. Tasks can be scaled up or down to handle varying workloads. Kafka Connect workers execute connectors and their tasks. Workers run in distributed mode (on multiple servers) to enable redundancy, similar to the Kafka brokers.

Among its many uses, Kafka Connect allows TRACE technical teams and developers to fine-tune the connectors to achieve desired performance characteristics, like low latency or high throughput, or even trade-off delivery and replication guarantees to meet specific performance requirements. Examples of these uses include the ingestion of databases, the automatic capturing of updates in tables, the reading of files from multiple locations, etc., into Kafka topics for further stream processing.

4.1.5 REST Proxy

The Kafka REST Proxy provides a set of RESTful APIs to a Kafka cluster, which allows other services or platform administrators to produce and consume messages, view the state of the cluster, and perform administrative actions without using the native Kafka protocol or clients. These APIs provide the ability to list, create and delete topics, retrieve information with respect to the partitioning of topics and the distribution of the partitions between available brokers, and produce or consume messages from a specific topic. Additionally, administrators are able to retrieve details about message offsets and information regarding the number of Kafka brokers, as well as to change the configuration of the cluster nodes.

4.1.6 Schema Registry

The Schema Registry [7] is a complementary module of the StreamHandler that allows users to explicitly define the format and structure of the messages exchanged through the Kafka topics to maintain data consistency and compatibility across different applications. To achieve this, it utilises the JSON Schema and JSON serialisation formats that can define the structure of the data, ensuring that all data producers and consumers will comply with the agreed data format.

4.1.7 UI and Monitoring

A web-based graphical user interface has been integrated into StreamHandler to provide platform administrators with valuable tools that allow cluster-wide configurations and an overview of the cluster health status.

The monitoring module is a stack of well-known open-source tools (Prometheus [8] and Grafana [9]), which continuously monitors data streams and system performance, providing real-time alerts and notifications on any issues or anomalies within the system. Grafana supports custom dashboards that offer customisable visualisations of the cluster, as well as the streamed data.

4.2 Cloud-based Big Data Management System

The TRACE CDMS (Cloud-Based Data Management System) is a centralised platform component that uses cloud computing technology to persistently store data over the network. In collaboration with the Interoperability Layer (Section 3) and the StreamHandler (Section 4.1), they create a complete data management, processing, and storage system designed to operate at scale. This system allows the TRACE platform to continuously collect multi-domain data from several sources. The CDMS is used to store data

from external sources that need to be retrieved at a later stage, such as information related to shipments, upcoming transportation schedules, environmental conditions, vehicle specifications, inventory catalogues, etc. In addition, other platform components use the CDMS as clients to persistently store information, such as their produced output, logs, and monitoring data.

To serve the above needs, the CDMS is designed with a wide range of features, including auto-scaling capabilities that adjust resources based on demand and elastic storage that dynamically allocates space as data volumes increase. Integration and interoperability are also critical features of the TRACE CDMS. A comprehensive API Layer ensures seamless integration with other platform components, enhancing the system's versatility and ensuring compatibility with various tools.

Security is a critical feature of the CDMS. For this purpose, advanced data encryption protocols are employed to protect data in transit and at rest. Furthermore, automated backups and disaster recovery solutions are used to ensure data integrity and availability, minimising downtime and potential data loss. Finally, user/service authentication and role-based authorisation are supported to help maintain a secure and organised access control system within the TRACE platform. Users and services are identified on the API Layer through unique credentials, while role-based authorisation mechanisms are responsible for granting access rights and permissions based on the user/service's role within the organisation. This approach ensures that CDMS clients have the minimum necessary access level, enhancing security by reducing the risk of unauthorised access to system data and operations.

4.2.1 Internal Architecture

The TRACE Cloud-Based Management System follows a highly modular architecture. It is comprised of small, independent services that communicate with each other to offer advanced storage functionalities. Figure 14 presents the internal architecture of the CDMS and its modules. These modules are divided into multiple layers, as described in the following subsections:

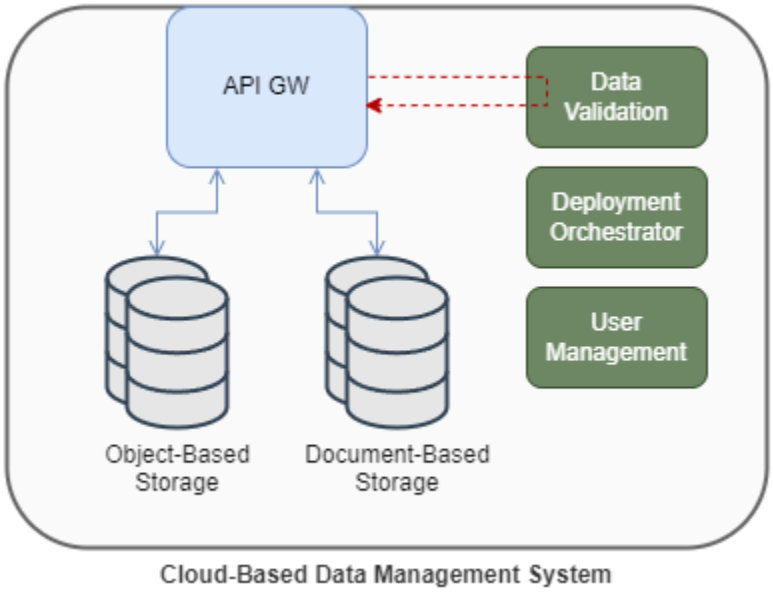

Figure 14: TRACE Cloud-Based Data Management System (CDMS) Internal Architecture

4.2.1.1 Storage Layer

The Storage Layer is the heart of the CDMS, which includes all the Database systems deployed to store data. As depicted in the diagram above, multiple instances of the databases are created to enable redundancy, fault tolerance and load distribution. For the needs of the TRACE platform, two main types of databases are used to store various kinds of data with different characteristics, namely, object storage and document storage. Depending on the type of data that each service needs to store, they can select one of the two available options as their backend storage solution.

The object storage database is responsible for storing data as objects, where each piece of information(object) is stored in a flat namespace accompanied by metadata and a unique identifier. The key feature of object storage is its ability to handle vast amounts of unstructured data, such as documents, images, videos, and backups, which are often better suited to being stored as discrete objects. Each object in an object storage database is stored with its metadata, which can be customised to include information about the data's origin, usage, and access policies, among other details. This metadata makes managing and retrieving data easier, as you can search based on these attributes. MinIO [10] was selected to implement the object-based storage of the CDMS. It is an open-source, high-performance object storage system that stores unstructured data such as photos, videos, log files, and backups as objects, where each

object is accessible via a unique identifier. MinIO provides the necessary features to manage, store, and retrieve large amounts of data efficiently, while supporting data encoding and versioning.

On the other hand, the document-based storage database is responsible for storing data in documents that include a key-value pair structure similar to JSON. Each document is a self-contained unit of data, which can contain complex data structures, including nested fields and arrays, which provides a high degree of flexibility when dealing with varying types of data. MongoDB [11] was procured to serve as the document-based database of the CDMS. It is a NoSQL database that operates on collections of documents, allowing other services to send/retrieve large volumes of unstructured or semi-structured data.

The following table provides more details on the stored data coming from various platform services:

Table 34: TRACE Storage Endpoints

Endpoint	Methods	Backend Storage
/api/v1/companies	Get	MongoDB
/api/v1/company/{id}	Get, Post, Put, Delete	MongoDB
/api/v1/areas	Get	MongoDB
/api/v1/area/{id}	Get, Post, Put, Delete	MongoDB
/api/v1/points	Get	MongoDB
/api/v1/point/{id}	Get, Post, Put, Delete	MongoDB
/api/v1/vehicles	Get	MongoDB
/api/v1/vehicle/{id}	Get, Post, Put, Delete	MongoDB
/api/v1/sensors	Get	MongoDB
/api/v1/sensor/{id}	Get, Post, Put, Delete	MongoDB
/api/v1/transportationmodes	Get	MongoDB
/api/v1/transportationmode/{id}	Get, Post, Put, Delete	MongoDB

Endpoint	Methods	Backend Storage
/api/v1/shipments	Get	MongoDB
/api/v1/shipment/{id}	Get, Post, Put, Delete	MongoDB

4.2.1.2 Interface Layer

The Interface Layer hosts an API Gateway (API GW), which acts as a middle-layer component between client platform services and the backend databases. It defines and implements all the necessary endpoints that will be exposed to the TRACE platform and is responsible for managing and orchestrating incoming API requests. Various RESTful API endpoints are created for data ingestion and retrieval, and, depending on the target URL, the API Gateway routes the client requests to the appropriate backend database. All the available endpoints, along with their details (such as the data schema, available methods, etc.), are documented on an OpenAPI specification [12] document and are exposed via a swagger UI [13].

Moreover, the API GW is used for user authentication/authorisation purposes, ensuring that only authorised clients can access specific URLs and databases. To achieve this, it is integrated with the user management module described in Section 4.2.1.3. to verify the credentials included in the API requests.

Finally, this layer hosts a data validation service that ensures that the received data follows the specified format and type and complies with the data schema presented in Section 3.2. This is a Python-based service that accepts batches of data and compares them with a pre-defined schema to ensure that they meet specific criteria for accuracy, completeness, format, and consistency. The API GW redirects the incoming data through the data validation service before forwarding them to the appropriate backend database. If the validation fails, the API GW will return a failing message to the client, producing the relevant logs.

4.2.1.3 Control Layer

The control layer acts as the brain of the CDMS, managing and coordinating the deployment and the interactions between the various system modules. It hosts two crucial modules: the deployment orchestrator and the user authentication.

The deployment orchestrator is a Python-based application responsible for coordinating the deployment process of the various modules comprising the CDMS. Through various configuration files, it i) automates

the deployment, management, and scaling of the DB instances; ii) manages the communication between the API GW, the user authentication module, the data validation module and the DBs; iii) continuously monitors the health of deployed modules.

The user management module creates a centralised point of control for managing service accounts and authentication for the CDMS. The platform administrators are responsible for creating dedicated client accounts and distributing the credentials for the services that will interact with the CDMS. Client services include these credentials on every REST call. Subsequently, the API GW that receives these requests checks with the user management module to ensure that the provided credentials are valid. Keycloak is the tool that has been selected for the implementation of the user management module. Keycloak is an open-source Identity and Access Management (IAM) solution that provides centralised authentication, authorisation, and user management services for applications and services. It is a flexible and highly customisable tool that supports easy integration with various other tools and programming languages.

5 Transport and Communication Infrastructure

Considering the layered TRACE solution architecture, the lower layer is defined as the infrastructure layer. This layer comprises the communication and transport infrastructure equipment/ vehicles/ devices etc. that are needed to provide the logistic services as well as to support the communication between the TRACE platform and the underlying logistics equipment. This section details on the TRACE infrastructure layer. It shall be noted that the infrastructural blocks (hardware or software) of this layer are necessary for the operation of the TRACE platform and the actual delivery of logistics services, (although not directly developed in the context of the project), thus analysed as part of the solution architecture.

5.1 Overall Architecture of Transport Infrastructure

Considering the transport infrastructure this includes the infrastructural elements that are needed for the actual delivery of goods i.e. Vehicles, type of roads, roadside equipment etc. as well as the communication devices and functions for the V2V (Vehicle-to-Vehicle) communication. Considering the TRACE Conceptual Architecture this corresponds to the “Vehicles” architectural components. Given that the transport infrastructure building blocks depend highly on the stakeholders involved in each logistics use case, and also given that the transportation infrastructure can be extremely versatile the following sections detail on these elements and the blocks on a per use case basis; namely for the Italian Use Case, the Slovenian Use Case and the Greek Use Cases.

5.1.1 Transport Infrastructure Layer for Italian Demonstrator

Transport Network Assets to Support Italian Use Case B

To ensure smooth execution for the Use Case B, the following infrastructure and assets are required:

MASA Infrastructure and Servers are essential to support connectivity and manage real-time data exchange between the vehicles, MASA, and TRACE.

At least two equipped Smart Bike that can connect and share information through the MASA infrastructure, enabling real-time delivery status updates to MASA and TRACE. A human operator will serve as the platoon lead, essential for testing platooning functionality.

The cycling lane in Modena (Diagonale Verde) must be available and a routing solution (provided by TRACE) for testing and for an efficient delivery the on-board parcels to test the delivery. To test the optional multi-modality delivering with drone, it is also necessary to have one UAV drone on a single cargo bike.

K*ey Vehicles*

- **Autonomous Cargo Bikes**: These bikes are equipped with sensors, GPS, and communication systems, allowing them to operate in a platooning. They are used for last-mile delivery services and can communicate with infrastructure systems (MASA and TRACE), enabling efficient navigation and coordination. (Capabilities details can be found in D4.1 [14])
- **Drones**: Capable of aerial navigation, drones are designed to deliver packages and small items. They use advanced sensors for obstacle detection and precise positioning, making them suitable for urban and rural environments (Capabilities details can be referenced in D4.1 [14]). Physical transport infrastructures are not needed for aerial navigation. Nevertheless, flight authorization is a relevant issue and predetermined flight corridors, or operational volumes should be defined with authorities and stakeholders to support drone logistics.

To have a safe scenario for the pilot demo B, the cycling lane, called “Diagonale Verde”, in Modena is needed. Drone flight does not require any specific infrastructure, given the possibility to rely on general purpose cellular communication; LTE/4G connection enables both remote piloting, thus supporting capabilities to handle more drones with limited pilots, that do not need to be deployed near the flight area, in the future, which allows scalability of this mode of transportation, when normative evolution will allow such operations (work in progress within JARUS).

The cellular network is also the baseline for all communications related to logistics; 4G will be used to interface both Drone and Lifebox to the MASA infrastructure through IoT (Internet of Things). It is also used to comply with evolving normative concerning the airspace rules specific to drones, i.e. the so-called U-Space. U-Space and operational authorizations on specific operational volumes can be considered a virtual infrastructure that defines a sort of allowed routes for drone logistics; this is a relevant issue to enable applicability of drone use cases, which are erroneously considered to perform a free route navigation, which is not feasible from a normative point of view.

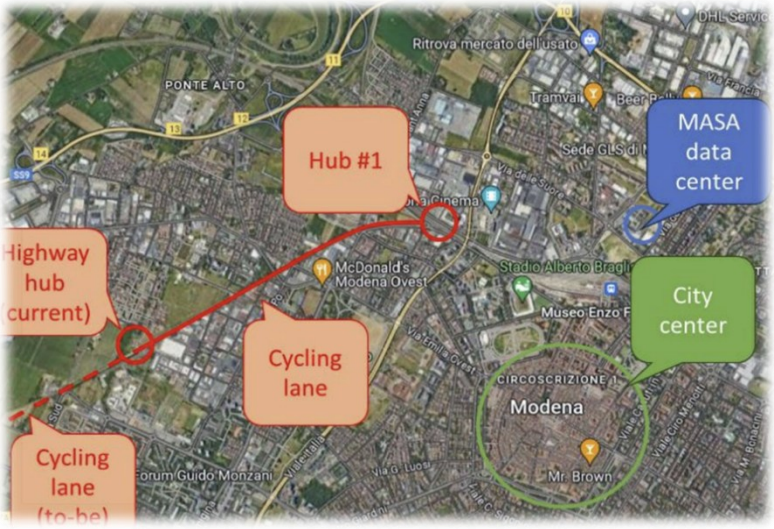

Figure 15: Cycling lane, called "diagonale verde", in Modena

5.1.2 Transport Infrastructure Layer for Slovenian Demonstrator

The Slovenian demonstration will showcase at least three different types of autonomous and non-autonomous sustainable vehicles, which will perform last-mile delivery and last-mile pickup operations in the AV Living Lab's ecosystem BTC City Ljubljana, based on TRACE Platform outputs.

Shipments will be exchanged between the Post Office (Pošta Slovenije), BTC Company's Logistic center premises and end business customers located in BTC City Ljubljana, to showcase multimodality/synchromodal operation between two logistic companies, using existing physical and network infrastructure.

Figure 16: Location of BTC City Ljubljana within the City of Ljubljana

5.1.2.1 Location of Slovenian demonstration

The Slovenian demonstration will take place in AV Living Lab's ecosystem of BTC City Ljubljana, the largest shopping, business, and leisure center in Slovenia, rich with content, people/visitors and traffic situations. It was chosen, as it can replicate situations in urban environments.

During the demonstration, existing road, bicycle and pedestrian infrastructure (roads, bike paths, sidewalks or pedestrian areas) will be utilized to replicate actual urban environments, where upgrade/altering existing infrastructure is often costly or impossible.

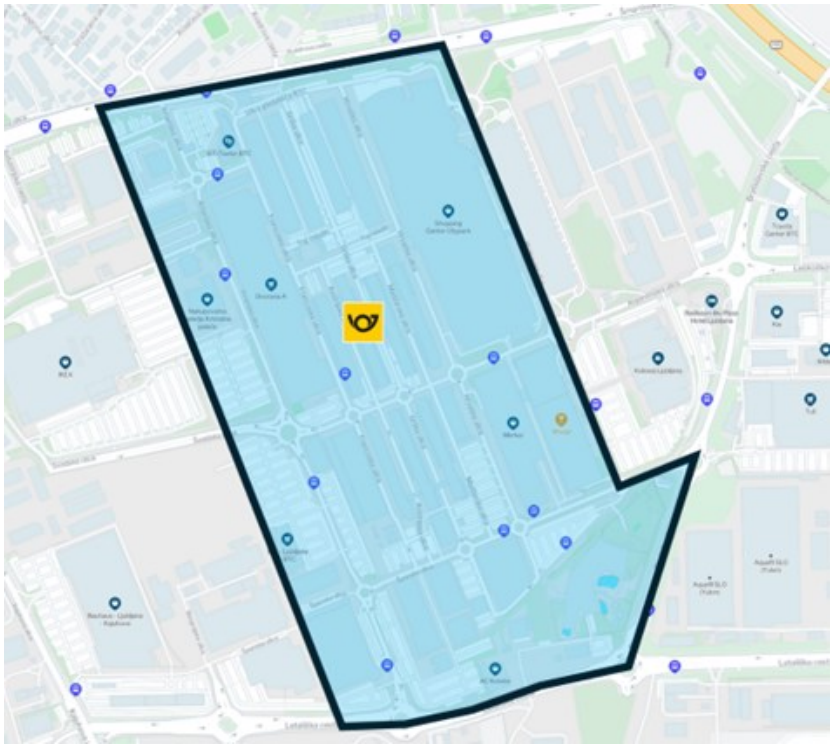

Figure 17: Map of Slovenian demonstration area

Post office of Pošta Slovenije at Avstrijska ulica 3 will act as a main logistic hub for demonstration activities, while BTC Company's logistic premises in the area will act as a Consolidation center.

5.1.2.2 Vehicles

At least three different means of transport will be used in Slovenian demonstration:

- Autonomous delivery robot
- Cargo bike
- Electric van

5.1.2.3 Autonomous delivery robots

PS will use one of the commercially available autonomous delivery robots as part of demonstrations and integration into TRACE platform. PS has chosen the French manufacturer Twinswheel (TW) [15] as the most suitable provider. For demonstration activities, PS has selected the M model, which will be equipped with multiple separate compartments to ensure secure delivery to end users. The specifications of the TW ADR (Twinswheel Autonomous Delivery Robots) are listed below:

- 4 driving, suspended and steering wheels

- 4 driving modes: radio control, remote 4G control, follow-me, autonomous on virtual roads
- Lidar sensors + 2D & 3D cameras + IMU + encoders + GNSS
- Position, brake, turn signal lighting
- Eyes and sounds
- Wi-Fi and 4G connectivity (excluding SIM card)
- Multi-locker box with touch screen
- box weight: 50 kg, capacity: 100 kg
- Total number of lockers: 6
- Teleoperation station (computer with 3 screens)
- Dimensions: length 1500 x width 690 x height 1370 mm,
- Weight empty: 150 kg
- Maximum speed: 12 km/h
- Autonomy: 15 km, exchangeable batteries
- Mapping mode: The Twinswheel autonomous delivery robot enables straightforward terrain mapping through advanced sensors, including 3D cameras, 2D and 3D LiDARs, and other sensor types. The data collected during terrain mapping is used to generate a spatial layout where the ADR can navigate. Pickup location points, linked to recipients' addresses, are entered for precise routing.
- Autonomous mode: With the provided web applications, the robot is able to move autonomously in an urban environment
- Remote control (non-autonomous mode): The robot enables real-time remote monitoring and control takeover in critical situations, corresponding to SAE Level 4 autonomy.
- API connectivity: API connectivity with various systems, such as postal information systems, TRACE platform, etc.

Figure 18: Twinswheel's medium size delivery robot

5.1.3 Transport Infrastructure Layer for Greek Demonstrator

The Greek demonstration will showcase at least three different scenarios with various types of autonomous and non-autonomous vehicles, which will perform End-to-End delivery and last-mile delivery operations in three locations in Greece as defined in TRACE Deliverable D2.4 [16].

Shipments will be exchanged between ACS and HT premises while last mile delivery in a campus environment will be also demonstrated. The aim is to showcase multimodality/synchromodal operation between two logistic companies, using existing physical and network infrastructure, achieving resilience, performance optimisation and autonomy in logistics services.

5.1.3.1 Vehicles

The Greek demonstrations will revolve around three locations (in Thessaloniki for UC – storyline 1, in Chalkis for UC-Storyline 2, and in Athens for UC-Storyline 3). Different means of transportation will be used in each of the demonstrations depending on the particular storyline/ scenario, namely:

- Cargo Trains
- Cargo Trucks and supportive systems
- Mobile Robots
- Autonomous vehicles/ Drones (as in previous demonstrators).

5.1.3.2 Cargo trucks and supportive systems

The trucks utilized by ACS for daily shipments for the corridors from Athens to Thessaloniki and vice versa fare **semi-trailer trucks**. These vehicles are integral to long-haul freight transport, combining a tractor unit with a semi-trailer. These type of trucks with the specifications provided below will be the ones that will be used for the Greek scenario of TRACE project. It shall be noted that, trucks and roller cages used in this Use Case pose no specific requirements to the transportation (road) infrastructure as they are (human driven) vehicles with typical dimensions.

Trucks' Capacity:

- **Maximum Gross Vehicle Weight (GVW)**: Up to 40 metric tons, depending on regional regulations.
- **Payload Capacity**: Approximately 24 to 26 metric tons.
- **Volume Capacity**: Around 90 m^3.

Figure 19: ACS utilized semi-trailer truck at ACS Athens Central hub

Roller Cages

- **Capacity per Truck**: Each truck carries 36 roller cages.
- **Capacity per roller cage:** Approximately 110 shipments

Roller Cage Security Specifications:

- Locking Mechanism: Locking bar or latch to secure items during transport.
- Label Holders: Often include areas for labelling or barcodes to identify contents.

Figure 20: ACS' utilized semitrailer truck loaded with roller cages containing shipments

Trucks and roller cages are equipped with the necessary sensors and communication interfaces to enable data collection and communication with the TRACE platform.

5.1.3.3 Rolling stock vehicles

The Athens-Thessaloniki railway line is a critical component of Greece's transportation network, linking the capital city of Athens to Thessaloniki, the nation's second-largest urban center. Stretching approximately 500 kilometers, the line connects diverse geographical regions. Recent modernization efforts have focused on electrification resulting in reduced travel times and enhanced efficiency.

The Athens-Thessaloniki railway line is a standard-gauge (1,435 mm) electrified railway. The line under 25 kV AC, 50 Hz electrification. The freight and passenger trains share the same track.

The freight rail transport is carried out using container platform wagons, "Rgss" type and "362" series. The transport takes place with containers 45', 40', 20' normally provided by the customer. Containers can be both pallet wide and high cube.

Maximum gross weight per container: 30,00 tn

For regular large customers the service defines customers that can provide at least 2 containers per destination.

Regarding the **Rgss 362** wagon the specifications according UIC are the following:

Length: 18.520 meters

Tare weight: 23.5 tn

Maximum gross weight: 56 tn

Loading length: 60 ‘

Number of axis: 4

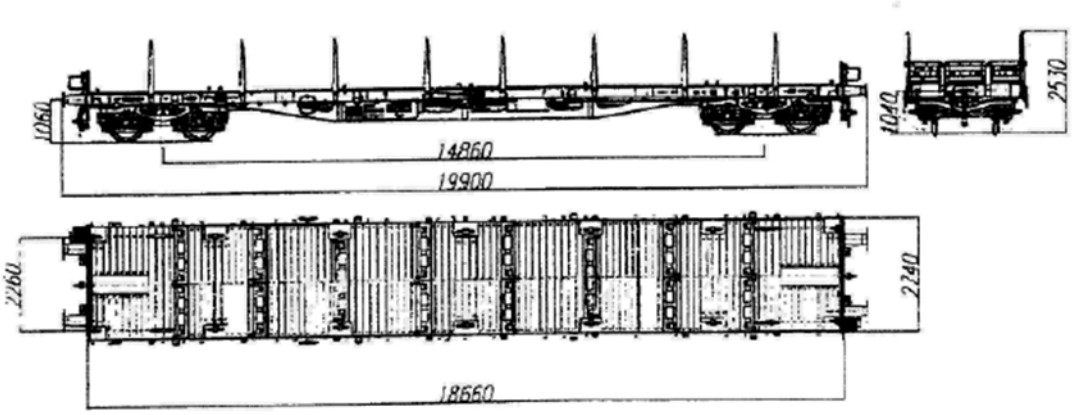

Figure 21: Rgss 362 wagon specifications

A train on the Athens-Thessaloniki route cannot carry more than 1200 gross tons or exceed a total wagon length of 600 meters. The primary locomotive used for this type of train is the Hellas Sprinter 120 electric engine.

Figure 22: Hellenic Trains locomotive

These locomotives weigh 80 tons and deliver 5,000 kW of power.

5.1.3.4 Mobile Robots

In this demonstrator, the RB-VOGUI (provided by Robotnik) will be used. It is a mobile robot designed to adapt to various transportation tasks, with a particular focus on outdoor environments and industrial applications such as agriculture, construction, logistics, and research projects. Thanks to its robust design, autonomous capabilities, and versatility, it is also ideal for last-mile deliveries in complex and challenging urban environments. Key characteristics of this transportation infrastructure/ vehicles are:

- **All-terrain omnidirectional kinematic base**: enables smooth mobility in both indoor and outdoor environments, as well as over complex terrains.
- **Teleoperation mode**: allows remote control of the robot for specific tasks.
- **Fully Autonomous Navigation**: the robot autonomously performs tasks, mapping its environment and making navigation decisions without human intervention.
- **Advanced autonomous navigation**: as it maps its surroundings and navigates using GPS, 2D and 3D LIDAR, and RGB cameras, ensuring safety with certified 2D safety LIDAR.
- **IMU sensor**: provides orientation and movement data for the robot.
- **GPS sensor**: enables precise outdoor localization.
- **3D LIDAR system**: maps and detects obstacles in the environment in real-time.
- **RGBD cameras**: capture color and depth images for recognition and navigation.
- **Certified 2D safety outdoor scanners**: ensure obstacle detection and safe operation.
- **Wi-Fi and Bluetooth connectivity**: enables versatile wireless communication, with compatibility for 4G/5G Wi-Fi.
- **USB, RJ45, and HDMI connectors**: provide options for data exchange and external device connections.

The RB-VOGUI combines cutting-edge technology and adaptability, standing out as an innovative solution for transportation tasks in both industrial and urban settings.

5.2 Communication Infrastructure Layer

Considering the telecommunication layer, the identified architectural blocks at the stakeholders' side include the end-user devices (sensors along with their controlling equipment), edge compute nodes hosting any near-to-end-user functions, and edge/mesh network equipment; while at TRACE platform side the communication infrastructure architectural blocks include the core network functions and the

underlying compute infrastructure along with any storage infrastructure. The sub-architecture of this layer depends highly on the selected communication technologies thus will be further detailed in the following sections on a per technology basis (i.e. 4G, 5G, WiFi). Considering the TRACE Conceptual Architecture this corresponds to the "Network Infrastructure" and the 4G5GCOM –cellular based solution- and WiFiCOM – WiFi based solution- architectural components.

5.2.1 Communication Infrastructure Layer Architecture

As already defined, the communication infrastructure layer can be based on various telecommunication network technologies/ deployments/ solutions ranging from cellular 3GPP to WiFi IEEE based. The selection of the underlying network layer will depend highly on the network deployment performance characteristics (achievable datarate, latency, availability – area coverage, mobility etc.). Three main network solutions will be explored for the TRACE deployment and Use Cases, namely: 3GPP 4G/LTE/5G NSA (non-Stand Alone), 3GPP 5G SA (Stand Alone) and IEEE WiFi; each adhering to the relevant standards' architectures.

5.2.1.1 5G Stand-Alone (5G SA) Network Architecture

The 5G network architecture comprises a set of segments namely:

- The 5G RAN (residing at the stakeholder's infrastructure block)
- The 5G Edge (residing at the stakeholder's infrastructure block but hosting TRACE Platform capabilities) and the 5G Core Network (5GCN) (residing at the TRACE platform block).
- The underlying transport network segments.
- The underlying compute infrastructure where the 5G RAN and 5G Edge and 5GCN components are hosted.

In particular, 5G Radio Access Network (RAN) is a critical component of the 5G telecommunications infrastructure that connects mobile devices to the core network through radio frequencies. It consists of base stations, antennas, and distributed network elements that enable high-speed, low-latency communication between devices and the network.

The 3GPP 5G core network provides the logical - session layer that is necessary for the transfer of communication services with specific QoS and mobility characteristics between the user (UE) and the Radio Network part and any internal or external the Data Network. An overview of the 5G Functional

Network Architecture as defined by 3GPP and mapped to TRACE functional architecture is illustrated in Figure 23.

According to 3GPP, the basic 3GPP network elements/ functions are the following:

- 5G/ New Generation - Radio Access Node (5G/NG-RAN or gNB)
- Access and Mobility Management Function (AMF)
- Session Management Function (SMF)
- User Plane Function (UPF)
- Unified Data Management (UDM)
- Authentication Server Function (AUSF)
- Network Repository Function (NRF)
- Policy Control function (PCF)
- Network Slice Selection Function (NSSF)

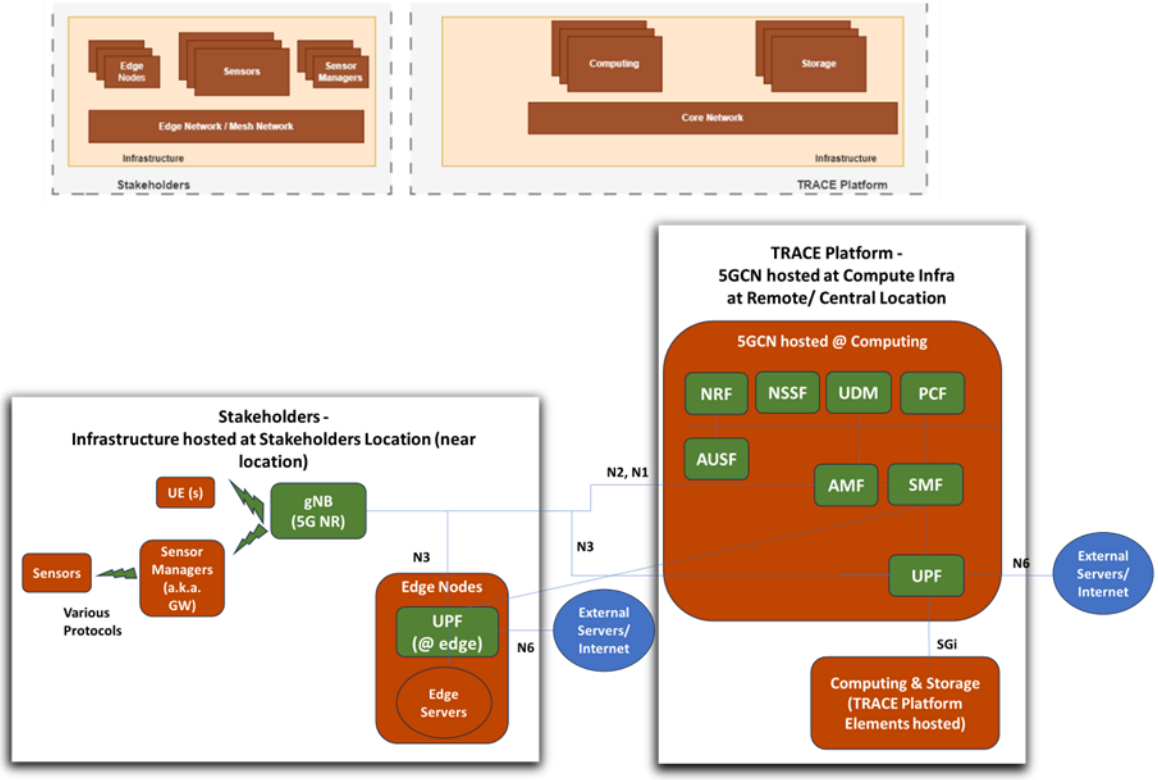

Figure 23: 3GPP 5G Network Architecture – mapped on TRACE Architecture.

The 5G/NG-RAN is built around the gNB (gNodeB), which serves as the central radio base station connecting user devices to the 5G core network. A gNB consists of three main functional components: the Centralized Unit (CU), the Distributed Unit (DU), and the Radio Unit (RU), which collaboratively handle processing, distribution, and transmission of data.

As far as the functionality of the basic 5G core functions is concerned, these functions support the following tasks/processes/capabilities specified by 3GPP 23.501:

The AMF function supports control plane functionality related to the access and mobility management of the communication session, and interfaces the UE via N1 and the NG-RAN via N2 interfaces. The key AMF functionalities are related to (i) registration and connection management, (ii) access, authentication, and authorization, including Network Slice-Specific Authentication and Authorization, and (iii) mobility management.

The SMF includes control plane functionality, and controls the Session Management, including the UE IP address allocation & management, the selection and control of the UPF of the session (via N4 interface), including configuration of traffic steering at UPF to route traffic to proper destination, establishment and release the tunnels between UPFs, N6-based forwarding, etc. It also determines the service and session continuity mode of a session.

The UDM supports: (i) subscription management, including the generation of 3GPP Authentication and Key Agreement (AKA) credentials, (ii) user identification handling, (iii) access authorization based on subscription data, (iv) storing of UEs serving NF information as part of the registration management (e.g., storing the serving AMF and SMF for UE's session), and (v) support to service/session continuity e.g., by keeping SMF assignment of ongoing sessions. To provide this functionality, the UDM uses subscription data that can be either stored in an internal repository or in an external one, namely the UDR (Unified Data Repository).

The AUSF supports authentication for 3GPP access and untrusted non-3GPP access.

The NRF supports the service discovery functionalities, especially related to the registration of NF-related information, e.g., NF and Service Communication Proxy (SCP) profiles, NF and SCP instances, health status of NF and SCP instances etc. Depending on the selected implementation of Network Slicing multiple NRFs can be deployed at various different levels, e.g., PLMN (Public Land Mobile Network) level (one NRF for each PLMN), shared-slice level (one NRF for a set of Network Slices), slice level (one NRF for each slice).

The PCF supports a unified policy framework to govern network behavior. It provides policy rules to Control Plane function(s) to enforce them, and accesses subscription information relevant for policy decisions in a Unified Data Repository (UDR).

The NSSF supports the selection of the set of Network Slice instances to serve the UE, and determines the AMF (or set of AMFs) to serve the UE. NSSF provides also support for Network Slice restriction-related tasks.

Regarding the 5G User Plane, the User plane function (UPF) serves as an anchor point for RAN and Intra-/Inter-RAT mobility, terminating the PDU session while it serves as an external PDU Session point of interconnect to the external Data Network (DN). It allocates the UE IP address and performs packet routing and forwarding, while enforcing UP policy rules e.g., traffic steering, QoS handling, etc. UPFs also perform transport-level packet marking in the uplink and downlink.

5.2.1.2 4G/LTE and 5G Non-Stand Alone network architecture

The 4G/LTE and 5G Non-Stand Alone network architecture comprises a set of segments namely:

- The 4G/LTE RAN, known as E-UTRAN (Evolved- UMTS Radio Access Network)
- The 5G NSA RAN (optional, extending the E-UTRAN)
- The LTE Core Network, known as EPC (Evolved Packet Core)
- The underlying transport network segments.
- The underlying compute infrastructure where the EPC components are hosted.

The 4G network architecture is built on a layered design that facilitates high-speed mobile communication, offering enhanced voice, data, and multimedia services. At its core, the architecture consists of three main components: the User Equipment (UE), the Evolved Universal Terrestrial Radio Access Network (E-UTRAN), and the Evolved Packet Core (EPC). The UE refers to the mobile devices used by end users, such as smartphones and tablets. The E-UTRAN provides the radio access network (RAN) and consists of eNodeBs (evolved NodeBs), which are the base stations that facilitate wireless communication with the UE. The EPC, on the other hand, is the backbone of the network that manages data routing, mobility, and connectivity to external networks like the internet.

The E-UTRAN plays a crucial role in 4G architecture by connecting the UE to the EPC. eNodeBs handle tasks such as air interface management, resource allocation, and mobility management, ensuring that users maintain seamless connectivity even when moving across different cells.

The EPC is a high-performance, IP-based core network that serves as the complete session controller of the 4G/LTE network. It consists of several components, including the Serving Gateway (SGW), Packet Data Network Gateway (PGW), the Mobility Management Entity (MME), the HSS (Home Subscriber Server) and the PCRF (Policy and Charging Rules Function). The HSS serves as the network subscriber database, where all network subscribers are listed along with their profile and authentication data. The MME, meanwhile, handles user authentication, session management, and mobility management, ensuring users stay connected while moving across different regions. The PCRF handles user session quality charging control.

The data/ user plane is handled by the SGW, which routes and forwards user data, while the PGW manages connectivity to external IP networks like the internet and ensures Quality of Service (QoS) for data sessions. Together, these components work to provide a highly efficient, low-latency, and scalable mobile broadband experience, enabling various applications. Edge resources can be exploitable for TRACE application components hosting by deploying multiple S/P-GWs near the stakeholders locations (e.g. at edge compute resources).

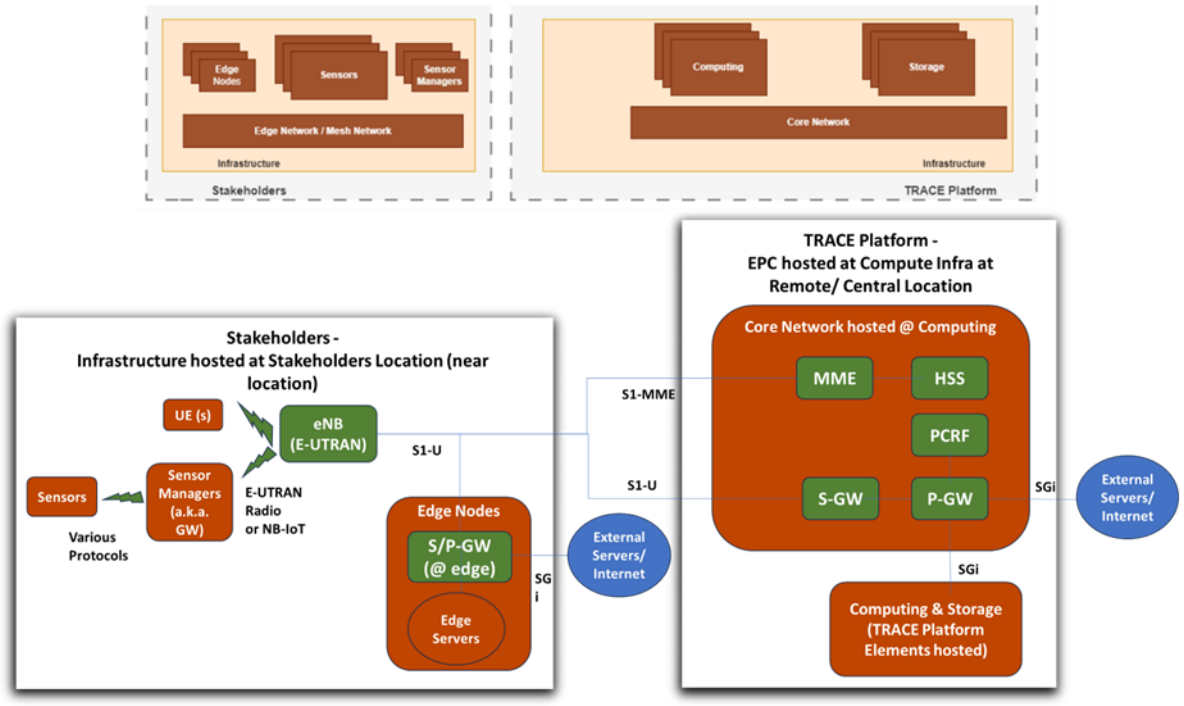

Figure 24: 4G/LTE Network Architecture – mapped on TRACE Architecture

The 4G/LTE network RAN can be extended with 5G New Radio nodes comprising the 5G Non-Stand-Alone flavour of 3GPP networks; essentially providing the data-rate enhancements over the 4G EPC. This architecture is illustrated in the following figure.

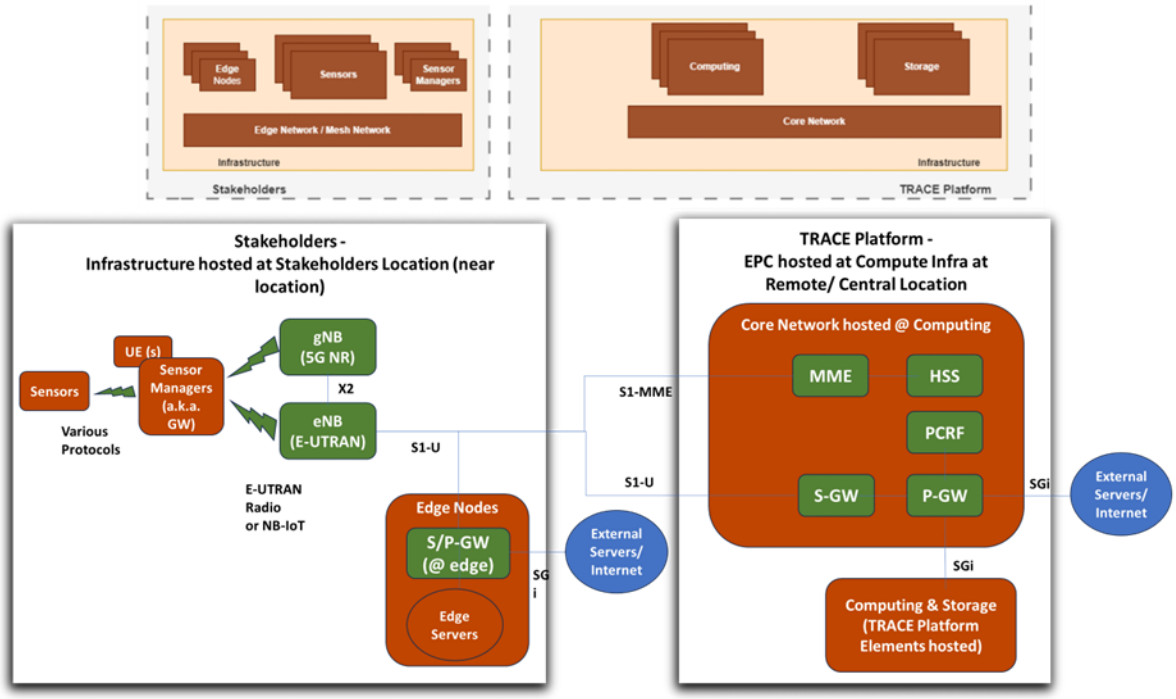

Figure 25: 5G NSA Network Architecture – mapped on TRACE Architecture

5.2.1.3 WiFi Network Architecture

IEEE 802.11x refers to a set of wireless networking standards under the broader IEEE 802.11 protocol family, enabling wireless local area network (WLAN) communication. The deriving network architecture consists of two main network elements the wired or wireless Access Points (APs), which act as intermediaries to connect clients (end-user devices) to a broader wired network or the internet and the Authentication Server (AAA – Authentication, Authorization and Accounting Server) which enforces security and access control policies when the APs are part of or target to access a commercial ISP (Internet Service Provider) network. The primary unit of the IEEE 802.11 architecture is the Basic Service Set (BSS), which consists of a single AP and its associated wireless devices. In more extensive setups, multiple BSSs can interconnect via a Distribution System (DS), forming an Extended Service Set (ESS). This design ensures scalable, seamless connectivity across multiple physical spaces.

The relevant standards incorporate advanced mechanisms to optimize performance, security, and reliability. For instance, Media Access Control (MAC) protocols regulate how devices share the wireless spectrum, mitigating collisions and interference. Additionally, IEEE 802.11x includes provisions for robust security protocols like WPA3, ensuring safe data exchange. Features like roaming, handled through the

ESS, enable devices to move between APs without disrupting connections. Modern iterations of IEEE 802.11x standards, such as 802.11ax (WiFi 6), introduce enhancements like orthogonal frequency-division multiple access (OFDMA) and target wake time (TWT), further improving efficiency and battery life in dense network environments. The selection of the flavour of WiFi network deployment depends highly on the applications/ services that need to be provided.

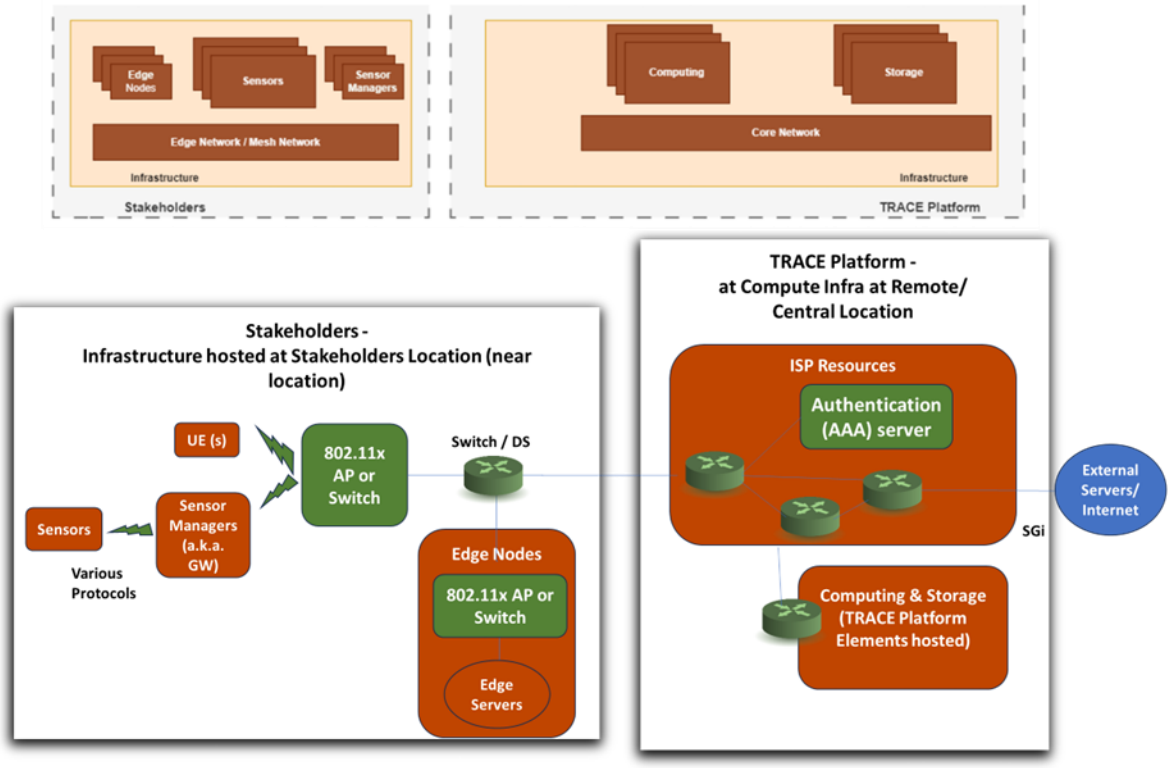

Figure 26: 5G IEEE 802.11x based Network Architecture – mapped on TRACE Architecture

5.2.1.4 Vehicle-to-Vehicle Network Architecture

For some services like platooning other network architectures are considered for V2V (Vehicle-to-Vehicle) communication, which are based on direct communication between vehicles of the same group without the mediation of the Infrastructure Network elements. Such network deployments are especially useful considering latency and cost aspects, as well as network traffic aspects. Such configurations are also useful considering areas with infrastructure network coverage holes.

For instance, in case of temporary lack of coverage, the group of vehicles will be able to perform the planned delivery and vehicles will be able to co-ordinate with each other. In cases where the leader of the

platoon detaches from the group temporarily (e.g. in order to make a delivery), the V2V module will be critical to communicate the updated configuration of the platoon, such as identifying the new leader and determining which bike to follow when multiple bikes are present. In this case, V2V communication—or potentially a combination of V2V and V2I (with Masa & Trace) communication—will ensure all actions are tracked and synchronized effectively.

In the context of TRACE, V2V communications are considered in the network infrastructure architecture, at the access network segment as shown in the following figure, while the protocols used can be versatile depending on the vehicle vendor, the vehicle/ logistics operator licensing (radio-frequency licensing schemes) etc.

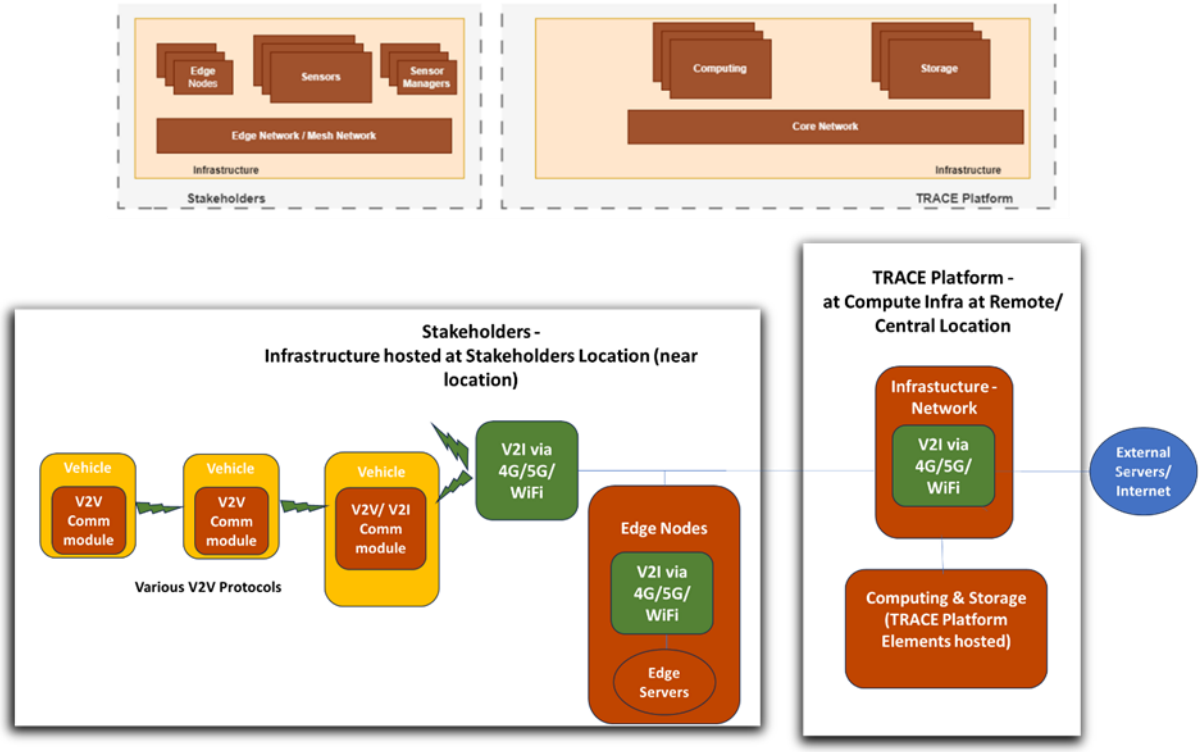

Figure 27: 5G V2V Communication – mapped on TRACE Architecture

5.2.2 Communication Infrastructure Layer for Italian Demonstrator

The network infrastructure layer that will support the Italian Use Case, will be based on 4G/5G network architecture/ deployment. The infrastructure that will be used for the communication system, involving

different transportation entities, including drones, cargo bikes, and platooning systems is illustrated in the following figure.

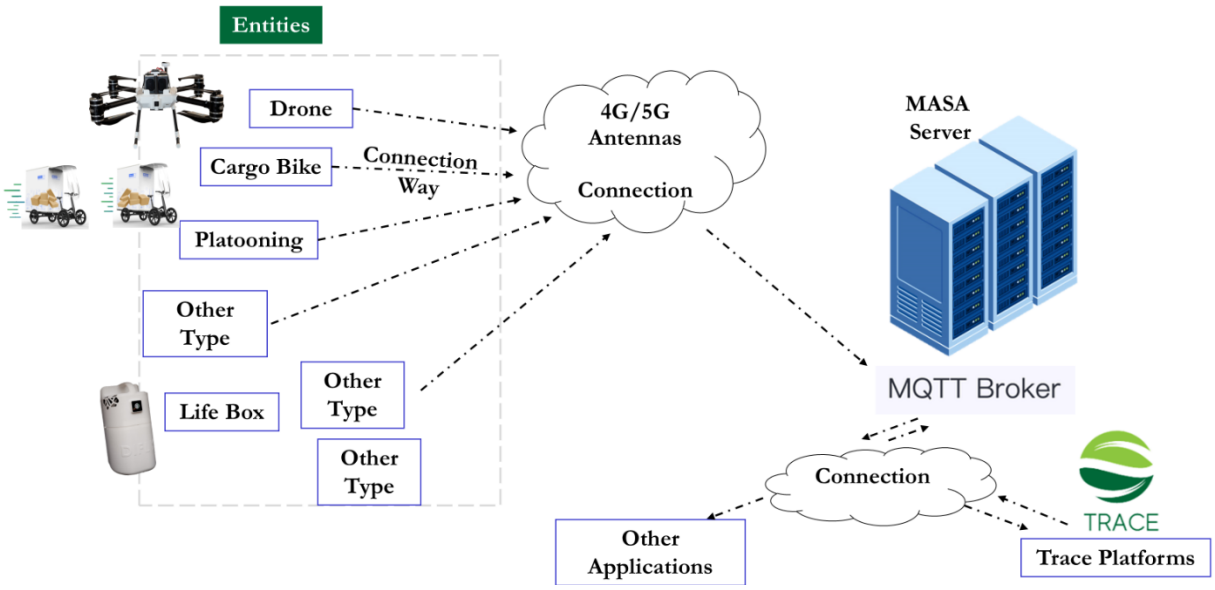

Figure 28: Communication Infrastructure Layer for Italian Demonstrator

In the context of this Use Case connectivity between the end-devices and vehicles and the MASA Server will be established over 4G/5G network infrastructure. The MASA server processes data, relaying it through an MQTT Broker that interface with the Trace Platform for further processing and applications. Additionally, roadside equipment or smart devices, represented as “Other Type”, can feed data into this system, ensuring integration between vehicles and the platform for real-time logistics tracking. Platooning based on V2V (TDMA – Time-Division-Multiple-Access based solution) communication will be also deployed.

5.2.3 Communication Infrastructure Layer for Slovenian Demonstrator

The network infrastructure layer that will support the Italian Use Case, will be based on 4G/5G and WiFi network architecture/ deployment. The infrastructure that will be used for the communication system, involving different transportation entities, including the autonomous delivery robots, the cargo bikes, and the electric delivery vehicles is illustrated in the following figure.

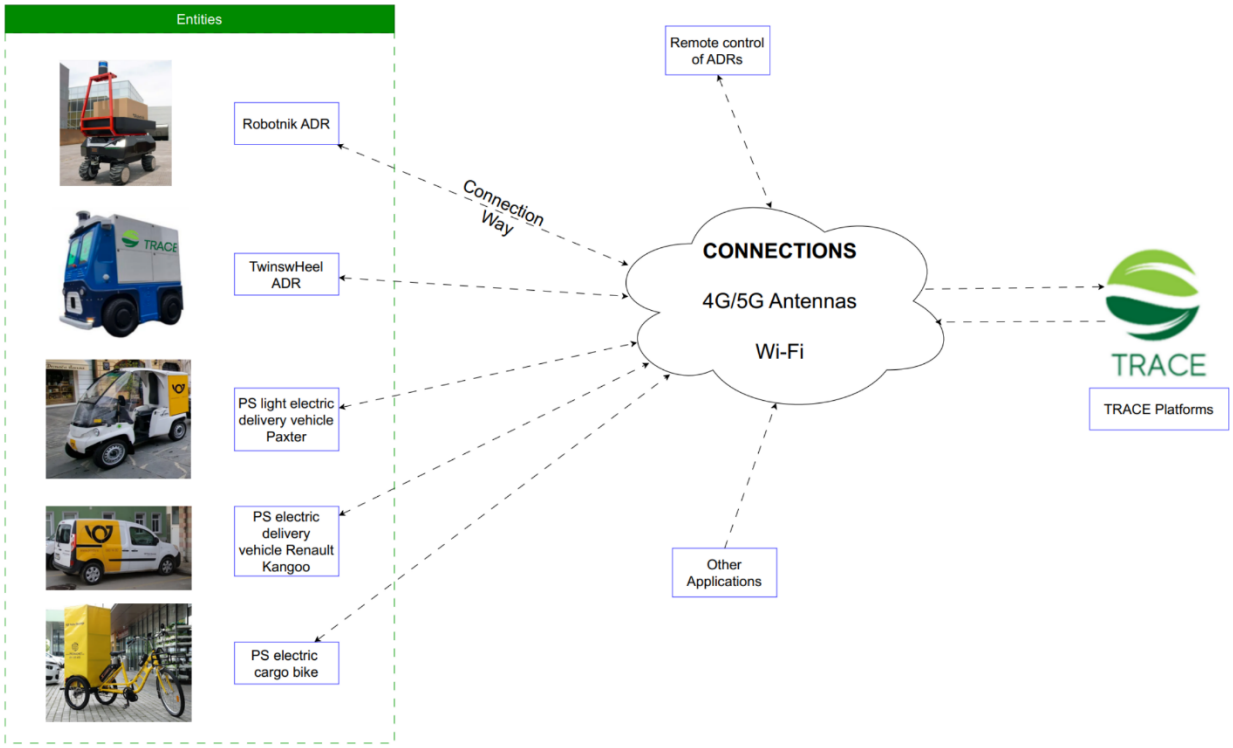

Figure 29: Communication Infrastructure Layer for Slovenian Demonstrator

5.2.4 Communication Infrastructure Layer for Greek Demonstrator

The network infrastructure layer that will support the Greek Use Cases, will be based on 4G/5G network architecture/ deployment, while WiFi connectivity can be also an option depending on the devices network interfaces. The infrastructure that will be used for the communication system, involving different transportation entities, including cargo trains, trucks and last mile vehicles (drones, autonomous vehicles), roller cages, etc. is illustrated in the following figure.

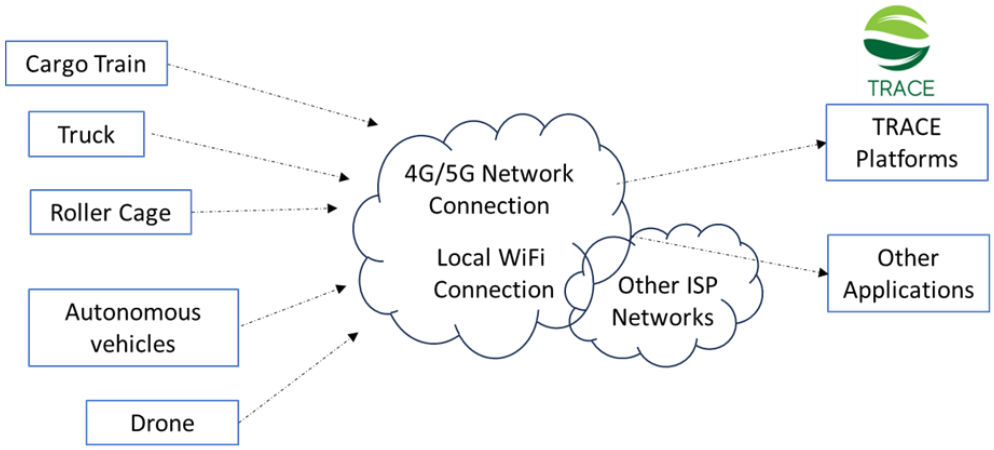

Figure 30: Communication Infrastructure Layer for Greek Demonstrator

The vehicles and end devices of this Use Case communicate via a 4G/5G network that is deployed at the ACS and selected HT station sites in Thessaloniki, Chalkis and Athens areas. This connectivity enables communication from the end devices to the NKUA Cloud Infrastructure where the TRACE Platform will reside for data processing and application provisioning. Connectivity between the devices or/and the TRACE Platform with external / other applications will be performed over the 4G/5G network and any interconnected ISP networks (DNs).

6 Conclusions

This report presents the comprehensive reference architecture for the TRACE platform, establishing a robust foundation for integrating diverse logistics operations. The architecture is structured in distinct layers, from infrastructure to user interface, with clearly defined components and integration points that enable seamless interaction between different systems and stakeholders.

The Interoperability Layer addresses a fundamental challenge in logistics by enabling diverse systems to communicate through a unified semantic framework, eliminating the need for data replication while maintaining data consistency across different schemas and terminologies. This is complemented by a sophisticated Big Data Management System, comprising the StreamHandler for real-time data processing and the Cloud-based Data Management System (CDMS) for persistent storage, ensuring efficient handling of large-scale logistics data.

The Transport and Communication Infrastructure layer demonstrates the platform's versatility through three distinct implementations across Italian, Slovenian, and Greek demonstration sites. Each implementation showcases different combinations of transportation assets (including autonomous delivery robots, drones, cargo bikes, and traditional vehicles) and communication technologies (4G/5G, WiFi, and V2V), proving the architecture's adaptability to various operational contexts.

The detailed documentation of integration points and interfaces ensures that future development and deployment of the TRACE platform can proceed systematically, while the modular design allows for scalability and the incorporation of new technologies as they emerge. This architecture sets the stage for the next phase of development, providing a clear roadmap for implementing an innovative solution that will enhance efficiency, sustainability, and collaboration in logistics operations.

The next major milestone will be the release of D3.2 (Report on reference architecture B) at M32 (January 20260, which will incorporate learnings from the initial implementation phase and provide updated specifications based on stakeholder feedback and demonstration results. This iterative approach will ensure that the TRACE platform remains aligned with user needs while maintaining its technical excellence and innovation potential.

7 References

[1] TRACE Consortium, "D2.1 - TRACE Technical Requirements (A)," 2024.

[2] TRACE Consortium, "D3.3 - TRACE platform (alpha release)," 2024.

[3] "VOWL plugin (github)," [Online]. Available: https://github.com/VisualDataWeb/ProtegeVOWL.

[4] "Kafka Apache Documentantion," [Online]. Available: https://kafka.apache.org/documentation/.

[5] "Apache Zookeeper," [Online]. Available: https://zookeeper.apache.org/.

[6] "Kafka Connect," [Online]. Available: https://docs.confluent.io/platform/current/connect/index.html.

[7] "Schema Registry," [Online]. Available: https://docs.confluent.io/platform/current/schema-registry/index.html.

[8] "Prometheus," [Online]. Available: https://prometheus.io/.

[9] "Grafana," [Online]. Available: https://grafana.com/.

[10] "minIO," [Online]. Available: https://min.io/product/overview.

[11] "MongoDB," [Online]. Available: https://www.mongodb.com/docs/manual/.

[12] "Swagger OpenAPI specification," [Online]. Available: https://swagger.io/specification/.

[13] "Swagger UI," [Online]. Available: https://swagger.io/tools/swagger-ui/.

[14] TRACE Consortium, "D4.1 – Infrastructural Elements of the TRACE Platform (A)," 2024.

[15] "Twinsweel," [Online]. Available: https://www.twinswheel.fr.

[16] TRACE Consortium, "D2.4 – Ecosystem Development, Safety and Use Cases (A)," 2024.

[17] "Kafka REST Proxy," [Online]. Available: https://docs.confluent.io/platform/current/kafka-rest/index.html.

[18] "Keycloak," [Online]. Available: https://www.keycloak.org/.

ANNEX A: TRACE Platform Components List

Task	Component's Name	Component's Short Name	Description	Deployment environment	Dependencies	Lead Partner
T3.2	Offline Data Query	ODQ	Offline copy of the query-able table of companies to prevent an overload of the companies database. This snapshot of the tables could be updated every "N" minutes. Data could be standardized by the common information needed/sharable between companies info	Not clear at the moment		UNIMORE
T3.2	TRACE Semantic Framework - Data Model	TSFDM	The common language used for data coming from logistic companies within TRACE. A network of interlinked RDF(S)/OWL ontologies to semantically represent data from logistics companies. Includes REST API to interact with the TRACE semantic framework	TRACE cloud, Deployment: Docker	Function 1 - Query Data Input: TRACE Cloud services, Output: Connectors Function 2 - Get data Input: Connectors, Output: TRACE Cloud services	NKUA
T3.2	Input Transformer	InTransform	Software to transform the incoming data into TRACE data modeling language	TRACE cloud, Deployment: Docker	Input source(s) [Logistics companies] --> This module (transformation) --> TRACE Processing Components	UGLA/UTH/ CERTH
T3.2	Output Transformer	OutTransform	Software to transform (homogenize) the TRACE results to the end user	TRACE cloud, Deployment: Docker	TRACE Processing Components --> This module (transformation) --> End Users	UGLA/UTH/ CERTH
T3.2	Graphical User Interface (GUI) for logistics companies	GUI	The GUI for interacting with logistics companies	Cloud	[GUI] <-> with all the TRACE components	UTH

Task	Component's Name	Component's Short Name	Description	Deployment environment	Dependencies	Lead Partner
T3.3	Cloud-based Data Management System	CDMS	A Cloud-based HA data lake that supports data storage, querying, and retrieval functionalities. It will be used to store data produced within the context of the TRACE platform, as well as information from external sources.	Cloud - Containers	[CDMS] --> RESTful APIs --> [API GW]	INTRA
T3.3	StreamHandler	SH	A distributed HA event-driven streaming platform that handles a large amount of heterogeneous data, messages and events, enabling the interconnection of all different data sources in the project. It makes feasible the real-time or near real-time streaming of events from multiple producers to multiple consumers.	Cloud (SH Core) On-premises (SH Connectors)	[SH] --> RESTful APIs --> [CDMS]: Persistent event storage [SH] --> TBD --> [T4.3 Events Mgmt components (TBD)] [T4.2 Infrastructural Interfaces and Communications (TBD)] --> TBD --> [SH]: Incoming streams of events	INTRA
T3.4 & T4.2	V2V low power communication System	V2VlpCOM	It is composed of n communication devices capable of transporting information (to be defined) between the different vehicles in proximity. This can take place either among bikes being part of a platoon or between bikes of different platoons depending on the stakeholders needs. This will allow for example the Main bike of a Platoon to collect all the data of the Cargo Bikes composing the platoon	Cargo Bikes	[V2VlpCOM] --> Mesh Networking --> [V22lpCOM] [V2VlpCOM] --> 2 be defined --> [On Vehicle data system] (to access cargo data, bike data …) [V2VcomSEC] -->must be integrated into --> [THIS COMPONENT]	CSEM
T3.4	4G/5G communication system	4G5GCOM	It constitutes the underlying (mobile / wireless) network infrastructure that is necessary for	Infrastructure (PUCKAI)	[4G5GCOM]--> Sink Manager	OTE & HT (for Greek demo),

Task	Component's Name	Component's Short Name	Description	Deployment environment	Dependencies	Lead Partner
			transferring the data from stakeholder resource terminals to the TRACE platform		[4G5GCOM]--> IDM [4G5GCOM]--> SHDM	MOD (for Italian demo). PS & AVLL (for Slovenian demo).
T3.4	WiFi communication system	WiFiCOM	It constitutes the underlying (wireless) network infrastructure that is necessary for transferring the data from stakeholder resource terminals to the TRACE platform	Infrastructure (PUCKAI)	[4G5GCOM]--> Sink Manager [4G5GCOM]--> IDM [4G5GCOM]--> SHDM	OTE & HT (for Greek demo), MOD (for Italian demo). PS & AVLL (for Slovenian demo).
T3.5	CI/CD Stack	CICDS	A robust CI/CD platform that automates the processes of code building, testing, and deployment. It supports seamless integration with version control systems, provides extensive automated testing frameworks, and enables automatic to manual deployments across different environments.	Cloud	[CICDS] --> Support the dev & testing --> [All TRACE software tools]	INTRA
T3.5	API Gateway	APIGW	A collection of backend services, acting as a reverse proxy to streamline API calls, aggregate data, and implement common functionalities.	Cloud	[APIGW] <--> RESTful <--> [Platform components (TBD)]	INTRA
T3.5	TRACE Lightweight Dashboard	DSB	A lightweight user-friendly interface designed to provide real-time insights and control over various aspects of the TRACE platform.	Cloud	[DSB] --> RESTful APIs --> [API GW]	INTRA
T3.5	Monitoring and Logging Infrastructure	MonLog	A distributed framework that continuously collects, analyzes, and visualizes data regarding the performance and health of applications and infrastructure	Cloud (MonLog core) Alongside other components	[MonLog agents] <--> TBD <--> [Platform components (TBD)]	INTRA

Task	Component's Name	Component's Short Name	Description	Deployment environment	Dependencies	Lead Partner
				(MonLog agents)		
T4.1	Low-Power 3D Scene Registration System for Enhanced Platoon Driver Awareness	VisionAI 3D scene reconstruction	We aim to develop a vision system for real-time 3D scene registration of the surrounding environment, with algorithms running directly on edge devices. The system will prioritize low power consumption and a compact form factor, making it ideal for integration into constrained environments like helmets, drones, bikes, etc. The core application will be on a motorcycle driver's helmet, where it will enable 3D reconstruction of the surrounding area in real-time, providing enhanced spatial awareness and safety. By utilizing efficient algorithms and lightweight hardware, the system will operate on minimal power while maintaining the high performance required for accurate scene registration. This technology aims to improve driver decision-making through precise environmental mapping, supporting smoother and safer maneuvering.	At CSEM by installation on a helmet. Later on, integration with the bike system.	Interaction with other sensing systems, with cargo bikes BCB (Bike Connection Box)	CSEM
T4.1	SUM-X MY 2024	Cargo Bike	Cargo Bike with 2 m^3 transportation capacity and 250kg. FormFactor: 4-wheecycles. Engine: Bafang M410 HD. Equipped with actuators. 3 types of actuators related 3 different functions: Self Steering, Drive Forward, Self-Braking, Braking, Emergency stop included in the scope of the project Actually no reverse gear has been included in the scope of the project. The actuation system is intended to be connected with self-drive system from UNIMORE. www.sumbicycles.com	Open road, Closed area for self driving test. data sharing with UNIMORE's driving system	Receiving data from Self Driving System. Sharing protocol to be clarified and agreed.	OLV-UNIMORE
T4.1	Bike Connection Box	BCB	The autonomous bike vehicle, need the integration of sensors capable of seamless communication with an onboard system. These sensors would encompass one or more IP-graded camera	At UNIMORE and test it on bikes in the same	[THIS COMPONENT] --> [Interacts with the Sensors] --> [May be interact with others	UNIMORE

Task	Component's Name	Component's Short Name	Description	Deployment environment	Dependencies	Lead Partner
			mounted at the front of the cargo bike for precise localization using fiducial markers situated in the front and on the back of each Cargo Bike . GPS and Gyroscope sensors are also mandatory in order to have the exact rotation and position of the Cargo Bike in the environment. Furthermore, for effective Vehicle to Vehicle (V2V) communication, we establishing a local network shared among vehicles through WiFi antennas. This setup would serve as a vital safety mechanism, enabling the platoon to halt promptly in unsafe conditions. The same WiFi antennas can also be used to interchange information with the external infrastructure (V2I). The entire setup must be waterproof. Microphone could be also be useful for further feature.	deployment environment.	BCB (Bike Connection Box)] [THIS COMPONENT] --> [Interacts with external infrastructures like MASA (already GDPR compliant) and forward to the TRACE Platform?]	
T4.1	Smart Bike Autonomous Driving System	SBADS	Autonomous Software Stack capable of handle and move a sensorised Cargo Bike with its attuation mechanisms. The AS stack need an Nvidia High Performance board (e.g., Jetson Nano or Xavier NX) with waterproof enclosure.	At UNIMORE and test it on bikes in the same deployment environment.	[THIS COMPONENT] --> [Interacts with the Bike Connection Box (BCB)] --> [May be interact with others Cargo Bike] [THIS COMPONENT] --> [Interacts with the Bike Connection Box (BCB)] --> [Interacts with the Cargo Bike attuation System]	UNIMORE
T4.1	Robotnik Automation unmanned ground vehicle model RB-VOGUI	RB-VOGUI	Within the scope of the TRACE project, the RB-VOGUI includes 4 500 W traction motors with brakes and another 4 steering motors that allow omnidirectional movement. As for safety, it includes 2 2D safety laser scanners and also includes 3 emergency buttons, 2 of them fixed, on the front and back of the robot, and a third remote safety button, which will be used for testing. The	The RB-VOGUI has been tested in open terrain and crops; within the framework of	Navigation inputs required. There are different options available, point to point using GPS signal, using a ROS based planner or moving the robot by performing speed	ROBOTNIK

Task	Component's Name	Component's Short Name	Description	Deployment environment	Dependencies	Lead Partner
			robot also has other sensors for location and navigation: RGB front camera, 3D laser sensors or GPS are the most relevant. 5G and WiFi connectivity can be enabled. The maximum payload is 150 kg and can reach up to 2.5 m/s (9 km/h), but the maximum recommended speed is 1 m/s (3 km/h).	the project it will circulate in closed areas, roads and/or sidewalks. The driving system is in the process of being defined. The robot software runs on the on-board computer and can be connected to a server to exchange the necessary information, both sending data for monitoring and receiving action commands.	control if an external driving system is used.	
T4.1	Optimisation module	-	Software for feeding information to the optimization algorithms regarding TRACE system's vehicles	Not clear at the moment	Input source(s) [Unknown at the moment] --> This module (filtering/formatting) --	CERTH

Task	Component's Name	Component's Short Name	Description	Deployment environment	Dependencies	Lead Partner
					> Optimization algorithms [T4.4, T4.3?]	
T4.1	Sink Manager component		Northbound and Southbound interface for equipment used in the Greek pilot	On-board the transportation means (each node)	Input: sensors, Output: TRACE back-end	NKUA
T4.1	DIFLY Xplora2 UAS		A small Unmanned Aerial System (MTOM<25kg) capable of carrying 4 kg of Payload. The main intended payload is the LifeBox. The drone allows adding transportation segments that do not require dedicated infrastructures (roads, bike lanes). Facilities are required for deployment, storage and maintenance, possibly located not far from start or end locations of the delivery. Cellular coverage supports the sharing of all information with IoT.	The UAS is handled at DIFLY facilities. It will be deployed on the field for the demonstration activities. Connection to the cloud for information sharing is independent of aircraft real position and flight status.	Stakeholders: hospital and local authorities for use case details. Sharing logistics information from/to TRACE network.	DIFLY
T4.1	DIFLY LifeBox		It is a component that can be carried by any vehicle or means of transportation (even hand carried). It is the payload for biomedical applications. It protects and assures temperature and environmental parameters for the medical payload. Parameters are monitored and sent to IoT. It also tracks the position and communicates with logistic platforms, regardless of the vehicle.	Developed at DIFLY premises, to be used in experimental trials.	Stakeholders: hospital and local authorities for use case details. Sharing logistics information from/to TRACE network.	DIFLY
T4.2	RB-VOGUI interface	RB-VOGUI interface	Some information like robot state, battery state, safety state, process status, GPS position, ... must	There are different	Devices or modules to be communicated with.	ROBOTNIK

Task	Component's Name	Component's Short Name	Description	Deployment environment	Dependencies	Lead Partner
			be shared with the platform. Other kind of information could be shared (frontal camera images, ...) Action commands to control the robotic platform (joint speeds, navigation plans, ...)	communication interfaces available. The robot is running ROS and ros (topics/services/actions) can be used to take sensors information and ask for actions. This ROS node runs on the robot's integrated computer. Due to the modular architecture of ROS other interfaces can be implemented ex, Kafka or MQTT interfaces.	Information to be shared.	
T4.2	StreamHandler Data Management (it's more a contribution	StreamHDM	Software components for managing data from the TRACE interoperability layer to process in the stream handler. The component will provide an interface to receive data to the streamhandler. It	Docker containers	[TraceInteroperabilityLayer] -> StreamHDM	UGLA

Task	Component's Name	Component's Short Name	Description	Deployment environment	Dependencies	Lead Partner
	than a component)		will also provide functionality to store streamhandler output to TRACE storage			
T4.2	Kafka Managers (it's more a contribution than a component)		Software to integrate data from infrastructures	Docker/ containarised env		NKUA
T4.3/ T4.4	Resources Monitoring and Events Manager	REManager	It monitors the data coming from sensors and the events management module and triggers the Scheduler to provide new allocations/routes when events are present or new data arrive	Cloud	[StreamHandler] -> [REManager] -> [Scheduler]	UTH/UNISYSTEMS
T4.3	Event Management Module	EMModule	This component receives input from MOModule and makes a decision on if an event is present adopting a rule base.	Server/Cloud	[StreamHandler]-> [MOModule]-> [EMModule]-> [MIManager]-> [Scheduler]-> [MOModule]	UTH
T4.3	Monitoring Module	MOModule	It is connected with the StreamHandler and gets sensory and other data and detects any deviations from the transfer plans delivered by the Scheduler and Route Optimizer	Cloud	[StreamHandler]-> [MOModule]-> [EMModule]-> [MIManager]-> [Scheduler]-> [MOModule]	UTH
T4.3	Fleet Monitoring Manager	FMManager	It monitors the location of all available vehicles that are active in a specific range of area and can be used in case of re-allocations in real time	Cloud	[StreamHandler]-> [MOModule]-> [FMManager]-> [MIManager]-> [Scheduler]-> [MOModule] [StreamHandler]->[FMManager]-> [MIManager]-	NKUA

Task	Component's Name	Component's Short Name	Description	Deployment environment	Dependencies	Lead Partner
					>[Scheduler]-> [MOModule]	
T4.3	Mitigation Manager	MIManager	It is connected with the Event Management Module to apply the appropriate mitigation plan, e.g., trigger the Scheduler for re-allocations by sending the affected shipments and the available vehicles. Another example: the component could decide that no re-allocation is required.	Cloud	[StreamHandler]-> [MOModule]-> [EMModule]-> [MIManager]-> [Scheduler]-> [MOModule]	UTH, CERTH
T4.4	Scheduler	Scheduler	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.	Cloud	[Scheduler] -> Interoperability Layer -> [Scheduler] User Interface -> [Scheduler] -> User Interface [Scheduler] -> Trace Storage -> [Scheduler] [Scheduler] -> External Sources -> [Scheduler] [Scheduler] -> Block Chain -> [Scheduler] [V2VcomSEC]->must be integrated into->[THIS COMPONENT]	UTH/TUC/CERTH

Task	Component's Name	Component's Short Name	Description	Deployment environment	Dependencies	Lead Partner
T4.4	Route Optmizer	RouteOpt	This component is responsible for calculating the shortest paths with their estimated travel time, and distance.	Cloud	[Scheduler] -> [RouteOpt] -> [Scheduler]	TUC/CERTH /UTH/UNISYSTEMS
T4.5	Distributed Ledger Structure	DLT	Mechanisms to securely connect with the Algorand Blockchain	TRACE System		BC5/SIDROCO/UGLA
T4.5	Digital Wallet	SSID	A system that enables the interface to securely store digital identity credentials.	Interface		BC5/SIDROCO/UGLA
T4.5	PKI Ecosystem	PKI	A system that facilitates the integration of Public Key Infrastructure (PKI) into existing platforms.	Decentralized Ledger		BC5/SIDROCO/UGLA
T4.5	Smart Contract	SC	This component will handle all transactions related to authentication and storage of references on the blockchain. It serves as the backbone of the system, facilitating seamless communication between all involved components. It is responsible for executing the PKI ecosystem and ensuring the unshakable integrity of collaboration agreements.	Blockchain	[Scheduler] -> [BCInterfacce] -> [BlockChain]	BC5/SIDROCO/UGLA
T4.6	V2V com security	V2VcomSEC	A security layer to be integrated into the V2V communication protocol AND a secure protocol allowing for an exchange of keys in an ad-hoc set of nodes belonging to the same platform (e.g., all vehicles of the same logistics company).	Cargo Bikes	[V2VcomSEC] --> must be integrated into --> [V2VIpCOM]	CSEM
T4.6	Secure boot and firmware update for the V2V communication nodes	V2VcomBOOT	A system that allows the microcontrollers implementing the secure V2V communication functionality to verify the software upon boot, and an update in the form of a (set of) binary images to be transformed into secured update packages, deliver these to the V2V com MCUs, such that the latter can verify the update and apply it securely.	Cargo Bikes	[V2VcomBOOT] --> must be integrated into the same device as --> [V2VIpCOM]	CSEM
T4.6	Intrusion Detection Module and	IDM	A classifier will be developed upon the streams of sensors / communications to detect potential intrusion and fire mitigation actions. The module will be integrated with the other components of	On pilots' site (alongside the Sink Manager) and	[IDM] -> must be integrated into the same device as the sensors,	UNIS (lead) SID

Task	Component's Name	Component's Short Name	Description	Deployment environment	Dependencies	Lead Partner
	Events Management		the system to facilitate the management of recovery activities when intrusion is detected.	on the TRACE cloud platform - Docker container	communication infrastructure and the other components adopted for authentication and data preservation	
T4.6	Data encryption for the TRACE platform storage	DataStoreCrypt	Content encryption for TRACE user data in storage (distributed or centralized), ensuring the data is accessible only to the authorized parties.		[DataStoreCrypt] -> integration with TRACE platform is needed, so that the platform can retrieve decryption secrets correctly, whenever needed.	SID
T4.7	Virtual Reality application for real-time remote monitoring of smart vehicles through the TRACE platform	Virtual Cockpit	The Virtual Cockpit is a virtual reality application that will offer the following functionalities: 1) 3D map route visualization and current coordinates, 2) Event visualization through real-time stream connection, 3) Emergency functionality (outbound)	Meta Appstore / Cloud	A head-mounted display is required. Optionally controllers for the hands or an external color camera.	CDW/NKUA