

New <u>IC</u>T infrastructure and reference architecture to support <u>O</u>perations in future PI Logistics <u>NET</u>works

D2.5 PI networking, routing, shipping and encapsulation layer algorithms and services Final

Grant Agreement No	769119	Acronym	ICONET	
Full Title	New <u>IC</u> T infrastructure and reference architecture to support <u>O</u> perations in future PI Logistics <u>NET</u> works			
Start Date	01/09/2018	Duration 30 months		
Project URL	https://www.iconetproject.eu/			
Deliverable	D2.5 PI networking, routing, shipping and encapsulation layer algorithms and services - final			
Work Package	WP2			
Contractual due date	30.09.2020	Actual submissio date	on 30.09.2020	
Nature	R	Dissemination Level	I PU	
Lead Beneficiary	VLTN			
Responsible Author	Kostas Zavitsas, VLTN			
Contributions from	IBM, CLMS, ILS, NGS,	CNIT		

Document Summary Information



This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No 769119.

Disclaimer

The content of the publication herein is the sole responsibility of the publishers and it does not necessarily represent the views expressed by the European Commission or its services.

While the information contained in the documents is believed to be accurate, the authors(s) or any other participant in the ICONET consortium make no warranty of any kind with regard to this material including, but not limited to the implied warranties of merchantability and fitness for a particular purpose.

Neither the ICONET Consortium nor any of its members, their officers, employees or agents shall be responsible or liable in negligence or otherwise howsoever in respect of any inaccuracy or omission herein.

Without derogating from the generality of the foregoing neither the ICONET Consortium nor any of its members, their officers, employees or agents shall be liable for any direct or indirect or consequential loss or damage caused by or arising from any information advice or inaccuracy or omission herein.

Copyright message

© ICONET Consortium, 2018-2021. This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both. Reproduction is authorised provided the source is acknowledged.

TABLE OF CONTENTS

E>	kecutive	Summary	8
1	Intro	duction	9
	1.1	Changes since previous versions of the report1	0
	1.2	Processes, Algorithms and services for PI shipping, encapsulation networking and routin 11	g
	1.3	Deliverable Overview and Report Structure1	1
2	Back	ground reading and summary of relevant iconet work1	2
	2.1	Overview of the Transport Process	2
	2.1.1	Main actors and roles in Transport and Logistics1	2
	2.1.2	Breakdown of processes for transporting goods via Pl1	3
	2.2	The ICONET layers	4
	2.2.1	The ICONET Services	4
	2.3	GPICS components description 1	5
	2.4	PI Architecture	8
	2.4.1	Generic PI1	8
	2.4.2	PI business specific processes	9
3 Shipping service			
	3.1	Shipping Service Considerations 2	1
	3.1.1	Order Conceptual Layer 2	1
	3.1.2	Transport Conceptual Layer 2	2
	3.2	Shipping Service Design	2
	3.2.1	Collection Phase 2	2
	3.2.2	Design Phase 2	3
	3.2.3	Initialization Phase 2	4
	3.2.4	Iterative Phase	4
	3.2.5	Real Time Updates and Decisions 2	5
	3.3	Shipping Service Implementation 2	6
	3.4	Service sample application and design guidelines 2	8
	3.4.1	Generic Shipping Service implementation sequence2	8
	3.4.2	Shipping Service in the PoC2	9
	3.4.3	Design guidelines	0
4	Encap	osulation Service	2
	4.1	Encapsulation Service Considerations Overview	2
	4.1.1	PI-containers	3

	4.1.2	-	Encapsulations layers	33
	4.2	En	capsulation Services	36
	4.2.1		Constraints	36
	4.2.2	-	Encapsulation and Consolidation Protocols	37
	4.3	Alg	gorithmic Approach	38
	4.3.1	•	D3 bin packing algorithm	38
	4.4	Sei	rvice sample application and design guidelines	41
	4.4.1		PI Hub operations and encapsulation optimisation	41
	4.4.2	-	Design guidelines	42
5	Netw	vork	king service	43
	5.1	Ne	etworking Service Considerations	44
	5.1.1	•	Representation of logistical network features	44
	5.1.2	-	Data Structure	49
	5.1.3		The Transportation Planning Manager	51
	5.2	Ne	etworking Service Protocol	52
	5.2.1	•	Network discovery module	52
	5.2.2	-	Data packaging module	53
	5.2.3		Optimal fulfilment store assignment to orders	55
	5.3	Sei	rvice sample application and design guidelines	55
	5.3.1	-	TEN-T PI Network	56
	5.3.2	-	eCommerce	57
	5.3.3		Design guidelines	57
6	Rout	ing	Service	60
	6.1	Ro	uting Service Considerations and Design	61
	6.1.1	-	Computational complexity of optimal routing algorithms	61
	6.2	Ro	uting Service Protocol	65
	6.2.1		Service Design	65
	6.2.2	-	Routing Protocol Communications Flow	70
7	Discu	ussio	on and Conclusions	72
8	Refe	rend	ces	75

List of Figures

Figure 2.1: The three flows in a supply chain	12
Figure 2.2: Visualising the steps of transporting via PI	14
Figure 2.3: Representation of ICONET layers and services with respect to the OSI, OLI, and NC)Ll layers 15
Figure 2.4: GPICS three level structure of PI Hubs (ICONET D1.8)	16
Figure 2.5: Direct and indirect PI Container Routes	17
Figure 2.6: Transport process variations in the PI design (based on Sharraj et al., 2014)	19
Figure 3.1: Interactions between Layers/Components	20
Figure 3.2: Simple visualization of different phase	22
Figure 3.3: PI Order design	23
Figure 3.4: Shipment initialisation	24
Figure 3.5: iterative protocol upon PI container arrival at PI Node	25
Figure 3.6: Response to real time information acquisition by IoT	26
Figure 3.7: States managed by the Shipping Protocol	29
Figure 4.1: PI containers	33
Figure 4.2: PI encapsulation layers	33
Figure 4.3: A T-container filled with H-containers	34
Figure 4.4: H-container proposed in the Modulushca project	34
Figure 4.5: Several sizes of T-containers proposed in the literature	36
Figure 5.1: PI hub as a queuing system	45
Figure 5.2: A hub consisting of different queues	46
Figure 5.3: The PI represented as a queue network	46
Figure 5.4: A PI queue network with 4 π -hubs	47
Figure 5.5: PI hub transhipment operations representation	49
Figure 5.6: PI data structure classification	50
Figure 5.7: Networking service network discovery (Stage 1)	53
Figure 5.8: Interactions of Networking Service data packaging tool	53
Figure 5.9: North-South Europe PI Corridor	56
Figure 5.10: TENT PI Links and their weights	56
Figure 5.11: eCommerce PI Node information structure	57
Figure 5.12: PI Networking design recommendations	59
Figure 6.1: Nearest neighbour algorithm	62
Figure 6.2: Improve tour segmentation algorithm	63

Figure 6.3: Greedy algorithm	. 63
Figure 6.4: Optimality gap	. 64
Figure 6.5: Deep Learning based Routing Framework	. 65
Figure 6.6: Attention mechanism. Image source (Kool et al., 2019)	. 67
Figure 6.7: Integration of routing service with others in ICONET	. 71

List of Tables

Table 2.1: Reference to ICONET D1.8	15
Table 3.1: Shipping Service functionality Inputs/ outputs	27
Table 6.1: Greedy vs Nearest Neighbour (NN) heuristic	63
Table 6.2 Comparison of optimization strategies for NN	64
Table 6.3: Benchmarking for starting nodes and repetitions	64
Table 6.4 Performance of Graph ConvNet against exact and heuristic solutions	70

Abbreviation / Term	Description
ERP	Enterprise Resource Planning System
IP	Internet Protocol
LES	Logistics Execution System
NOLI	New Open Logistics Interconnection
OLI	Open Logistics Interconnection
PI	Physical Internet
РоС	Proof of Concept
RIP	Routing Information Protocol
WMS	Warehouse Management System

Glossary of terms and abbreviations used

EXECUTIVE SUMMARY

The report presents the Physical Internet (PI) four core services, namely the Shipping, Encapsulation, Routing and Networking. The services have been designed to align with the OLI/ NOLI (and ICONET) layers enabling a standardised approach to the PI implementation. The Generic Physical Internet Case Study (GPICS) has also been taken into consideration, as it offers the underlying PI ontology with which the PI Services have been designed to interact. The ontology of PI Links, Nodes and Services in its generic form is inherited from the GPICS and extensions are considered as each service is examined into further detail. The services are designed to account for various business types where various use cases arise. The sequence of communications of the PI Services design has been considered for each use case, aiming to enable the development of modular and robust services. The various applications contexts of the PI have also been taken into account, drawing on the adaptations of the four core services to the ICONET Living Lab requirements.

The description of the core PI Services, builds on the second version of this report and provides the final and complete service design.

The shipping service has an overarching manager role and can be divided into: design; initialization; arrival at PI node; and real time update modules. The function of the first two modules is associated with the request of shipment through the and the development of the PI Order. The Arrival at PI node module handles the sequential hops of PI containers in their route to their destination, while the real time updates module, communications with the IoT platform and collects data to track the performance of the PI shipment against its contractual obligations.

The encapsulation service investigates the bin packing algorithm as well as algorithms for overcoming its computational complexity. The encapsulation service addresses the encapsulation of cargo into PI containers, into H containers into T containers into PI Movers/Means. It offers a generic tool for improving operational efficiency and decision making at PI Hubs. Variations of the generic model are described that can also be utilised for efficient eCommerce encapsulation.

The networking service primary function is network discovery, in order to provide a standardised and complete representation of the PI for further decision making. Using the GPICS as a guideline, an enhanced data structure is proposed breaking down the PI network information into static and dynamic data for PI Links, PI Hubs and PI Movers. Furthermore, considering the ICONET Living Labs, several network representation approaches are considered focusing at varying network aggregations. A guideline for networking service implementation into different contexts is also provided.

The routing service investigates the computational complexity and heuristics for improving the solution time for generic and specialised PI routing problems. The performance of methods utilising an integration of optimisation and machine learning means are discussed in detail, before proposing an PI.

All Services presented in this report have been integrated with the PoC Platform. The communication between the PoC has been implemented by using either direct (Service X to Service Y) or indirect (Service X – Simulation – Service Y) Application Programming Interfaces (API).

1 INTRODUCTION

Goods and products are transported from one location to another where they will become more useful and valuable. There are several stages in the development of a product, starting from one or more raw materials, going through several processing and assembly stages, before it finally reaches a retail store, and is purchased by the final customer. In economies, agglomeration contributes to the expansion of supply chains as processes tend to be increasingly specialised. Additionally, products are becoming increasingly varied and complex emphasizing the need for supply chain flexibility.

For most developed countries, economic productivity (typically expressed through the Gross Domestic Product) is found to be directly associated to the amount of goods movement. Establishing an efficient system for moving goods, is an essential milestone for commerce while at the same time extracting higher capacity from legacy infrastructure such as railways, and motorways. Furthermore, with sustainability becoming an increasing concern, logistical solutions in transport became more relevant, aiming to satisfy transportation demand in an environmentally friendly manner. Although methods and technologies for planning and executing transport and logistics have improved with time, the main principles and inefficiencies still apply today.

Performance of freight transportation is one of the crucial elements for the sustainability of logistics and supply chain. The costs for the freight transportation can reach up to 60% of the total logistics costs for shippers, Collignon (2016) and inefficiencies in transportation costs can be characterized by economic, social and environmental inefficiencies and unsustainability. Despite efforts by transport companies, the frequency of empty trips remains high and average truck fill-rate is low. Overall, according to Eurostat (2017), at total transport level, most trucks in Europe fell in the range between 15 % and 30 % empty journeys. Moreover, freight transportation (in developed countries) is responsible for nearly 15% of greenhouse gas emissions. This ratio has been increasing despite ambitious reduction targets. Improved transportation efficiency is therefore an important objective of the Physical Internet and it aims to reduce logistics costs by building on concepts from Digital Internet that enabled the development of global system of the data transport across heterogenous networks exploiting standard datagrams and protocols.

The Physical Internet (PI) promises to revolutionise how transport and logistics is practiced, and to improve on critical variables such as cost, utilisation rates, and emissions through improved multimodal integration and open accessibility to static and mobile infrastructure. The core constraints, objectives and business processes involved in planning, coordinating and executing the transport of goods from origin to destination remain largely unaltered in a PI approach. What changes under the PI is the standardisation and interoperability of transport, logistics systems and processes. For these features of the PI to materialise, several information and decision support systems as well as standardisation and integration services require to be introduced. In this report we discuss the following transport and logistics processes under a PI approach:

- Encapsulation: Standardises the packaging process of cargo and goods that are consolidated/deconsolidated into π -containers for transportation via the PI. It is also responsible for the consolidation/ deconsolidation of π -containers into π -movers.
- *Shipping*: Specifies what has to be transported as well as the transportation process conditions and constraints. It is responsible to make appropriate adjustments to the shipping instructions to ensure compliance.

- *Networking*: Networking defines the interconnected infrastructure of available processing, storage and transporting facilities (transport services, terminals, distribution centres, warehouses) through which the goods will be transported from their origins (manufacturing, distribution and other locations) towards their customer(s) locations.
- *Routing*: Routing is a process that creates a plan that describes the stage by stage detailed visiting and usage of networking nodes and links from origin to destination.

The above processes:

- Have different planning horizons: long medium and short term
- Take place at different stages, have different durations and may be repeated in the space of a single transport process.
- Are supported by different IT systems (ERP, Transportation Planning, warehousing management, and others).

Each process receives requirements/constraints about one of the parties involved in the transport/logistics chain and makes decisions which are further communicated to and interpreted by other parties. In the context of PI these processes are further distributed and decentralised, since PI is a network of networks and each of the interconnected networks may be owned/controlled by different actors.

1.1 Changes since previous versions of the report

Version 1 of this deliverable provided a succinct and abstract definition of the shipping, networking and routing processes under PI. With analysis of real transport processes from the Project's Living Labs and the development of a Generalised PI model (GPICs) in Work Package 1, description of the PI processes was further elaborated.

Version 2 contained a more extensive description of PI processes as well as an early prototype of a route planner under PI. The Encapsulation PI process that requires specialised loading units (PI-containers) and other suitable handling and transporting equipment was described in detail. Protocols integrated with Operations Research algorithms for undertaking the encapsulation process were proposed. information aspects of encapsulation have been discussed in the first version of this report and also in deliverables D2.1 PI Reference Architecture v1 and D1.11 PI Protocol Stack and enabling networking technologies v2.

In the current and final version of the document, further case studies of PI shipping networking and routing are presented with reference to the Project's Living Labs and further shipping, networking and routing demonstrators are provided. Specific case studies derived from the Living Labs provide information for the elaboration of the PI processes.

1.2 Processes, Algorithms and services for PI shipping, encapsulation networking and routing

In this report we analyse the processes of transportation under a PI approach. The automation of such processes requires suitable IT systems and services. Such services are software implementations of planning, routing etc., decision support algorithms. There are already many types of systems (Transport Management Systems-TMS, Enterprise Resource Planning Systems-ERP, Warehouse Management Systems-WMS and others) that provide such services to transportation users and performers. In this report we attempt to analyse the processes from a PI perspective, to establish where new types of decisions need to be supported, new information services to support such decisions, and new enterprise systems to be developed or current ones to be overhauled.

Proposed PI Services utilise mathematical programming for optimising decision making for the generic shipment of cargo through the PI, as well as for various Use Cases that arise in the transport process. An algorithmic approach and variations of optimisation problems (changes in variables or constraints) are discussed were appropriate. Further algorithms are specified as PI models are applied to the Project's Living Lab scenarios, and prototype IT services implementing such algorithms, are envisaged in the course of the Project.

1.3 Deliverable Overview and Report Structure

The report consists of the following sections. Section 2 provides a background overview of the main actors and activities involved in the transport (end to end) process, the description of the OLI/NOLI layers, the GPICS representation and data structure. Using the interconnections made and background material presented in Section 2, the report then is decomposed into its parts: shipping, networking and routing, that are extensively discussed in sections 3-6 respectively. For each PI Service, the main considerations for the development of the service are discussed and analysed before proposing a generic protocol design. Each Chapter then, looks further into the detail of the components of each protocol for addressing both the core PI functionality, but also for addressing additional Living Lab inspired use cases. Examples of PI Service implementation are also provided illustrating Service functionality. In section 7 a summary of the work undertaken and concluding remarks are made that focus on the complexity that can be handled by the proposed PI services and their intercommunication. Section 8 contains reference resources.

2 BACKGROUND READING AND SUMMARY OF RELEVANT ICONET WORK

This section attempts to connect the various ICONET services to supply chain and logistics concepts and to align development work across the ICONET project and the literature in general. The Chapter builds on work presented in the previous two deliverables D2.3 and D2.4 on PI Services, as well as D1.8 on GPICS and D2.2 on the PI reference Architecture. Section 2.1 covers the transport process in practice, and attempts to make associations to the digital internet in order to establish the physical internet's functionality requirements. Section 2.2 focuses on the OLI, NOLI layers and connects the notions to ICONET layers and the architecture presented in D2.2, that is later used as the basis for developing the PI Services protocols. Finally, Section 2.3, focuses on the GPICS and the proposed generic network representation, that PI Services build on.

2.1 Overview of the Transport Process

2.1.1 Main actors and roles in Transport and Logistics

Logistics and transport involve the coordinating effort of several organisations, each of them focusing on a different part of the logistics and transportation process. A supply chain includes not only the manufacturer and the suppliers, but also transporters, warehouses, retailers, and even customers themselves. Although this may include organisations that have only an indirect role such as for example banks and insurance companies, we focus only on those organisational roles that are directly involved in the transport and logistics process.

Their involvement as stakeholders in the transport and logistics processes can be due to them owning (initially or ultimately- i.e. as sellers and buyers), the goods that are transported, the equipment and other resources by which the goods will be processed and transported, or because they are coordinators of the different processes and activities involved. We define different *roles* for actors participating in transport. It is possible for the same organisation to assume multiple roles in the process. So an organisation (or more accurately different legal entities within it) can at the same time be all seller and buyer, freight forwarder and carrier.

The functioning of a supply chain involves three key flows – information, product and funds as illustrated in Figure 2.1. The goal when designing a supply chain is to structure the three flows in a way that meets customer needs in a cost effective manner (Chopra, 2019).



Figure 2.1: The three flows in a supply chain

Of course, under the PI approach previous centralised activities may become devolved amongst multiple parties. It is more likely that multiple organisations may be responsible for each of the following roles, for a single transport act, rather than the reverse, i.e. the same organisation assuming multiple roles. For instance, in many supply/transport chains, the seller may utilise the services of a single (or small group) of carriers under a single contract. Under PI, where it is more likely that different carriers may be used for the different legs of transport, such carriers may not be in direct contract with the seller (or even under subcontract with the seller's main carriers).

With all the above taken into account, the following roles are defined that will be taken by actors participating in the shipping, networking and routing processes discussed in the following sections of the report.

- Shipper: The organisation that initiates the transport process by submitting a transport request to the freight forwarder.
- Customer: The organisation that will receive the transported goods.
- Freight Forwarder: The organisation that plans and coordinates/oversees the overall transportation process.
- Carrier: The organisation responsible for the physical movement of the goods
- Logistics Service Provider: The organisation that provides services related to the transport of goods such as storage.
- Intermodal terminal (hub): The organisation that provides the services to re-route or re-load goods onto different transport means, as they are moved towards their final destination, for example by trans-shipping. The hub can also provide other services, similar to those of the Logistics Service Provider. An intermodal terminal for instance, loads and unloads containers and trailers to and from rail wagons for movement by rail and subsequent movement by road.

2.1.2 Breakdown of processes for transporting goods via PI

Actors require to coordinate their activities to successfully transport goods. As illustrated in **Figure 2.2**, the processes involved are summarised below:

- Shipping instruction: The transport process initiation activity where a seller (shipper) instructs a freight forwarder to ship its goods via PI. Shipping instructions are used for instructing shipping conditions such as size, volume, packaging, etc.
- Network design and network selection: This process requires the tactical planning and design of the transportation network. This can be the responsibility of the shipper organisation or the freight forwarder. Due to the characteristics of PI this will require collaboration (peering agreements) between π -hubs (intermodal terminals), carriers and other logistics service providers. This is therefore a tactical (long term) process. In contrast, network selection is the process of choosing amongst alternative network paths, based on the shipping instruction. In network selection the freight forwarder/transport planner decide (in association with, based on the shipping instruction received, the path(s) through the PI that the cargo will follow to its destination. This also includes the decision of how to split the cargo in multiple consignments that will be forwarded along multiple paths to the final destination.
- *Transport route planning:* Planning the transport route is also a collaborative based decision where out of the possible routes through the selected network, a suitable route is selected, based on the characteristics of the consignment and also on the state of the network (current and future) at the time of planning.

• *Transport Execution:* Transport execution includes the activities that physically move cargo towards its destination, possibly via intermediate steps ('legs') and involving switching between different logistics equipment and additional activities that for example store, bundle or unbundle cargo. The order at which steps are followed can be pre-defined or a result of re-planning after the completion of each step.



Figure 2.2: Visualising the steps of transporting via PI

2.2 The ICONET layers

A high-level description of the layers that the ICONET project relied upon as well as their association to the OLI and NOLI layers is presented in D2.2 – PI Reference Architecture. The deliverable introduces the ICONET layers in association to the PI Services that are described into further detail into this report.

2.2.1 The ICONET Services

The ICONET reference model indicates the relationship between the OLI, NOLI and ICONET layers, as well as the ICONET services to be used for technical implementation for the ICONET project.

A Physical service was deemed out of scope for further technical work, as it would require significant effort to synchronize physical actions with the corresponding operations described in the PI concept and the Physical Layer. Additionally, adoption of these PI-based physical actions would be difficult at this current stage.

Moreover, the functionalities and offerings of the Link layer presented in a theoretical manner above, are very close to the specifications offered by the Shipping layer, and as such, the decision was made to unify these functionalities in a single Shipping service which in conjunction with other services, will provide mechanisms for efficient and reliable shipping. As illustrated in Figure 2.3, the services where the ICONET project will focus to encompass the majority of relevant functionalities on are the following:

- Shipping
- Encapsulation
- Networking
- Routing
- Optimisation
- Logistics Web

OSI Layer	OLI Layer	NOLI Layer	ICONET Layer	Resulting Service	
Application	Logistics Web	Product	Logistics Web	Logistics Web	
Presentation	Encapsulation	Container	Encapsulation	Encapsulation	
Session	Shipping	pping Order Orde		Shipping	
Transport		Transport	Transport		
Network	Routing	Network	Routing	Routing	
	Network		Network	Network	
Data Link	Link	Link	Link	-	
Physical	Physical	Physical Handling	Physical	-	

Figure 2.3: Representation of ICONET layers and services with respect to the OSI, OLI, and NOLI layers

2.3 GPICS components description

The GPICS modelling components are designed to allow the composition of a generic PI network trough standard modelling elements. GPICS components capture all ICONET layers, starting from the low aggregation of infrastructure for the Physical and Links Layers. Through the appropriate configuration, these elements represent different types of supply chain flows. The structure of the generic model consists of the following main elements:

Table 2.1: Reference to ICONET D1.8

	GPIC structure
GPICS Container	Unit load manipulated, stored, moved and routed through the systems and infrastructures of the Physical Internet.
GPICS Node/Hub	Location specifically designed to carry out logistics and transport processes and activities on PI containers.
GPICS Transport	Moving element used to carry PI containers through the PI nodes/hubs.
GPICS Corridor	Connection between two PI Nodes/Hubs directly connected.
GPICS Route	Set of GPICS corridors which connect a GPICS Node origin and a GPICS Node destination.
GPICS Network	Set of containers, nodes, movers/transport, corridors, and routes.
GPICS Roles	Actors/Agents involved in the operation of the PI Network.

Each of those core GPIC components, can be further subdivided into features and properties that describe it. For example, GPICS Container information include:

Container ID

- Origin and Destination IDs
- Sender and Receiver IDs
- Delivery time window
- GPS coordinates/ location.

Similarly, for each PI hub a classification in terms of functionality is applied into Gateway, Source, Switch, etc. For each of those functionalities, information on the infrastructure constrained throughput is also stored. Furthermore, the GPICS bring together the requirements of the four ICONET Living Labs. Taking into account the specificities of each of them, the representation and description are made through the creation of a hierarchical structure and the dependency of the GPICS Hubs. More specifically, a three-level structure (due to the maximum levels required by LL) of HUBS has been defined. Therefore, when defining, each Generic HUB belongs to L1, L2 or L3, in the instantiation process for a specific generic definition of a case study. The dependency is based on a simple rule: a L2 Hub depends directly on a L1 Hub and a L3 Hub depends directly on a L2 Hub. Indirectly, a L3 Hub depends on the corresponding L1 Hub as illustrated in Figure 2.4.

Therefore, for PI Hubs the information maintained are:

- Node ID
- Level
- Directly connected nodes
- Functions and their throughput
- GPS coordinates.



Figure 2.4: GPICS three level structure of PI Hubs (ICONET D1.8)

As for every graph, in the PI as well, information on the performance of Links are essential for enabling the ability to move efficiently through a network. According to graph theory, network links can be associated to one or more weights that are in essence link properties. Weights can depend on static infrastructure properties, as for example, the throughput of a road Link is associated (among other factors) to the number of lanes available and the type of road link (e.g. local or motorway). Link weights can also be associated to dynamic link properties such as travel time, that depend on the amount of traffic and congestion levels on the link.

In the PI context, to perform the transport of cargo from an origin to a destination, implies that a transport between these two PI Hubs is potentially possible. To make a transport reality, this Link must be configured in at least one GPICS route with at least one GPICS Move/Transport, it must be

configured in that route and with parametrised stops in those two GPICS Hubs. The basic information that defines a GPICS Link is the following:

- Link ID
- Starting and Ending Node IDs
- Type
- Capacity
- Congestion
- Transit time

A sequence of links forms a route. In the PI, PI containers are typically requested to follow a sequence of movements, where the origin of the first movement is the container's origin and the destination of the last movement is the container's destination. A GPICS route is inherited as a property of GPICS Transport or Movers, that represents the means of transport used to carry containers through the network. A PI Mover is responsible for carrying a container for one or more links or for the entire route. It is worth noting that a GPICS Route can form of a single link if such an option is available and it is found to be optimal, or alternatively form as a sequence of links that requires bypassing PI Hubs as illustrated in Figure 2.5. GPICS Movers do not necessarily follow the same route, and therefore the indirect route can be fulfilled either by a single or several Movers. Therefore, similar to the GPICS Link information, it is essential to maintain information of GPICS Movers, as in transport and logistics, the efficiency of transporting cargo between two nodes is effected by the service provided by the Movers.



Figure 2.5: Direct and indirect PI Container Routes

Aside the information related to the operational efficiency aspect, Movers, being the handlers of PI containers are also classified in terms of functionality, similar to PI Nodes. Functionality information enable the filtering of PI Movers, such as container transit requirements are met. The basic information for GPICS Movers is:

- Mover ID
- Type
- Path ID
- Service Frequency
- Service Functionality
- Capacity
- Fill Rate

2.4 Pl Architecture

2.4.1 Generic PI

The GPICS classification of components and their properties does a great deal, in enabling the mapping of PI functionality required for realising cargoes transport. ICONET networking, routing, shipping and encapsulation services are therefore designed to manage and make decisions on the operation of each of those network components. Following, the ICONET Layers from top-down, and utilising the GPICS component definitions, the Logistics Web Layer, is responsible for retrieving from the sender information on:

- Origin and Destination IDs
- Sender and Receiver IDs
- Delivery time window.

The Encapsulation layer is then responsible for fitting the cargo to be transported into one or more PI Container as efficiently as possible, storing their ID and enabling their GPS tracking. This forms an order to move a PI Container from a specific Origin to a specific Destination within a specific time frame. The Shipping layer is responsible for assigning Transport/ Movers to an Order. To achieve this, it communicates the PI Origin and Destination node information to the Routing and Networking services, and at frequent intervals checks if the routing instructions provided by the Routing and Networking services deviate from the PI Order requirements. This can be due to uncertainty or changes in the state of the Network. If that is the case, the Shipping services initiates a request to update the routing instructions for the specific at a higher urgency.

The Networking service once it has received the Origin and Destination ID's as well as the delivery time window information, accumulates the static and dynamic information on PI Nodes, PI Routes and PI Movers available for fulfilling the PI Order. In doing so the networking service filters out the network components and services that do not satisfy the specific orders explicit or implicit requirements. Explicit requirements include for example cargo refrigeration, or transport mode specifications, while implicit requirement is for example respecting the order delivery time window. The networking service, that also collects up to date information on the network status, both for infrastructure and services, communicates it to the routing service, that is responsible for identifying optimal routing instructions for each PI container.

Additional, to this core functionality of the PI, there are add-on functionalities and decision making tools, that enable the PI's efficient operation that form part of the PI core services. For example, an additional function of the routing service in small networks is to determine the routing of the PI movers, aside the routing of the PI containers.

Depending on the capability and the information available to enable enhanced decision making, the PI process can vary. As illustrated in Figure 2.6, the PI process proposed by Sarraj (2013) starts with encapsulation and routing before entering the repetitive segment for transporting cargo through the PI network 'hops'. When cargo has arrived at the destination, the container is unloaded and the loop terminates. In contrast, if live network status is considered in the design, only encapsulation stage is undertaken at initiation, and the repetitive segment starts with the routing identification. Therefore, as a PI container containing cargo, arrives at a PI Hub, the route to the destination requires to be re-evaluated with the most recent network information, as the network status might have changes and there is a more efficient route to the destination.



Figure 2.6: Transport process variations in the PI design (based on Sharraj et al., 2014)

2.4.2 PI business specific processes

The PI network operation is highly complex, and robust functionality is required to address the variability in operational requirements as well as technical capability. The following chapters of this report look further into detail in Shipping, Encapsulation, Networking and Routing services, and their protocols that handle PI complexity.

GPICS and ICONET layers are used as a guideline in addressing the Services requirements throughout the remainder of this report. In an effort to untangle PI complexity, references are made to the implementation of the four services to ICONET living labs (LLs), and best-practice recommendations are made for adapting or extending the PI Services presented as required for their successful deployment.

The four core services have been co-created in association with the Living Lab business users who have provided guidance in the design of the Use Cases the Services have been designed to handle, as well as the components that enable either the extension of the PI to handle specific business requirements or functionality. For example, in the Warehouse as a Service, there are cases where the destination of the shipment is not known, while in the case of eCommerce the fulfilment stores from where the order originates need to be identified. The Services design elements presented in this report, that have been discussed with users, will be tested in upcoming phase III and outputs will be included in the final Living Lab reports.

3 SHIPPING SERVICE

The Shipping Service encapsulates the functions of the conceptual Shipping Layer, whose role in a PI enabled network environment is to:

- Enable the efficient and reliable shipping of (sets of) PI containers from shippers to final recipients
- Study the management of the procedures and protocols for configuring the quality of service
- Monitor, verify (acknowledge), adjourn, terminate and divert shipments in an end-to-end manner
- Leverage the IoT means of T2.3 in accordance to the Blockchain principles of T2.4 whenever and wherever possible.

In order to fulfil these goals, the Shipping service takes on the role of the overall orchestrator of the PI services. As such, the Shipping Service is responsible for receiving PI enabled orders and with the usage of the capabilities offered by other services, make appropriate decisions to ensure the delivery or handle the non-delivery of an order in an end-to-end manner.

To accommodate the aforementioned goals, the Shipping Layer can be further conceptually divided into the Order and Transport layers. In short, it is more useful to think of these layers as a conceptual guideline while acknowledging that a full technical implementation could potentially generate a new conceptual paradigm. As such, the Transport layer will handle all the communications & data exchanges needed for a set of PI containers to be transported through the PI network, while the Order layer will monitor & update the PI order state from initialization to termination. An outline of the interactions between the proposed conceptual layers and the data flows can be seen in **Figure 3.1** below.





3.1 Shipping Service Considerations

3.1.1 Order Conceptual Layer

The Order Layer can be broken down to subservices, expressing various functions that are performed in the flow of the PI logistics. To start, the order service receives the initial non PI compliant order and is responsible for mapping the data deriving from external ERP systems into a **PI order**, which contains all the necessary data and is represented through the PI data model. Essentially, the PI Order composer function divides and/or combines the initial non-PI compliant shipment into orders based on product constraints (temperature, humidity etc.) and/or shipping constraints (final destination, deadlines etc). These orders are then placed under a transaction. This transaction, which essentially is a group of orders, is in place to ensure that despite the composition/decomposition of orders that might happen through the transport lifecycle of an initial non-PI order, the stakeholders that have an interest in this order can have a central access point that encapsulates all the necessary information and at the same time providing the necessary information to be able to re-bundle the original order after the various sets of PI containers reach their final destination, either for pickup by the relevant parties or for last-mile transport. Another function needed for the order composition is responsible for gathering or composing and the necessary documents (at this stage, order dispatch notes), and analysing them to create the relevant product and shipping constraints. This constraints function will be responsible to pass the relevant data to the encapsulation services during the initial cost calculation and when the orders are physically composed, to ensure that the encapsulation process will take into consideration the varying conditions of the products contained in these orders. Additionally, the Order services are responsible for maintaining the state of the aforementioned orders and transactions. In case of an event which affects the order in any way, the information is propagated to the order service in order to update the status of the orders that it manages and notify interested parties. These events are generated from sensors residing in the PI containers, propagated through the IoT cloud platform and finally received by the PI platform and sent to the order layer. Finally, in a step towards fully automated and decentralized process, the order services will leverage the Blockchain Ledger, creating Smart Contracts, binding stakeholders and containing code that will validate the constraints analysed by the constraints function, so that every time a new event is broadcasted the Blockchain ledger can automatically use the new data to validate against these preconceived requirements. Overall, some key functions of the Order services and the Shipping Protocol can be summarized as follows:

- Order State management: The order service layer serves as a receiver for IoT based events in order to update & manage the state of transactions & orders during the whole duration of the end-to-end trip. Additionally, the management process utilizes the innovation offered by the Blockchain ledger in an automated & decentralized way to be able to validate the constraints of an established order and generate impeachment information as well as notify the relevant stakeholders
- **Constraints function:** Special conditions deriving from product based or order based needs of a stakeholder are recorded and propagated to the necessary components from this function. Notices for Special transportation needs e.g. min/max temperature, deadlines, sub-orders, special handling etc. are all generated here.
- **PI order composer:** The composer function is responsible for "transforming" external non-pi orders into **PI orders** and composing/these orders as needed using the common PI data model used across the PI platform and grouping them under "transactions", using data from the Web Logistics Layer and creating relevant Smart Contracts in the Blockchain Ledger

3.1.2 Transport Conceptual Layer

The main purpose of the Transport Layer is to receive sets of containers organized by the Encapsulation Layer, and manage the end-to-end trip from of the aforementioned sets from their initial starting location to their final ending location. Apart from a function that is required to generate the final shipping documents made necessary by legal constraints, this layer is responsible for managing the shipping state of the orders generated from the order layer. As the Order Layer is responsible for monitoring, updating and validating the overall transactions of the PI enabled orders, the Transportation Layer has the same responsibilities for the end-to-end trips of these orders. After the initial encapsulation of the items in PI containers, the relevant container Ids will be returned to the transport service. Apart from the association with the PI order, these ids will be forwarded along with the original order to the IoT cloud platform, which in turn will return an API key to be used for authentication/authorization in future connections and requests. Using data from the IoT cloud platform, the Transport Layer can propagate received events & information for time of departure, current location, as well as the final arrival of the PI containers. The Transport Layer is responsible for utilizing these events in order to make further decisions about actions needed regarding the shipment of goods and informing other layers.

3.2 Shipping Service Design

The Shipping Layer, communicating either directly or indirectly with all other OLI/NOLI Layers acts, in principle as an orchestrator comprised of subservices and routines that ensure that any order will follow a well-defined, visible, trackable and optimal lifecycle from its Origin to its Destination. The subservices and functionalities of the Shipping Layer can be best described via the phases that an order passes through.



Figure 3.2: Simple visualization of different phase

3.2.1 Collection Phase

The Shipping Layer initially receives the non-PI compliant order and, combining the data gathered from external ERP systems, maps it to a PI order, which contains the necessary information regarding the Sender, the Receiver as well as the product and shipping constraints to be applied to the order. Communicating with the Network Layer, the Shipping Layer validates the Origin and Destination of the order and transforms them into PI Nodes, following the PI model.

The PI Order composer function, gathers all available PI orders and groups them into transactions based on product constraints (temperature, humidity, light etc.) and shipping constraints (final destination, deadlines). These transactions, being essentially groups of orders with similar constraints, ensure that despite any possible composition/decomposition of orders that might occur during the

transport lifecycle of an initial non-PI order, the stakeholders interested in said order will have a centralized point of access that encapsulates all the necessary information and at the same time provides a clear view on how to re-bundle the original order after the various sets of PI containers reach their final destination, either for pickup or for last-mile transport.

More importantly, the grouped, into transactions, orders ensure the end-to-end optimization of the delivery of all products in terms of CO2 emissions, deadlines, usage of containers and/or movers, congestion and costs.

3.2.2 Design Phase

The Design Phase is executed either on regular time intervals (e.g. every evening for normal orders) or on an ad-hoc fashion (e.g. for expedited orders). After a set of orders has been grouped into transactions of similar constraints, the Shipping Layer communicates with the Network Layer in order to

- 1. validate the Destination and,
- 2. to gather information concerning the availability, sizes and types of containers in the Origin.

After "hand-picking" the required containers (e.g. those with refrigeration), the list of containers and the products to be shipped are propagated to the Encapsulation Layer that, executing a bin-packing algorithm, optimally distributes all available products into their containers. The encapsulated containers are then returned to the Shipping Layer that communicates with the Routing Layer to obtain the best routes to use. Finally, all the collected information in PI Model terms is sent back to the Blockchain Ledger, as a digital Bill of Lading, that will inform the stakeholders on the decided, theoretical costs and parameters and await for approval.



Design

Figure 3.3: PI Order design

3.2.3 Initialization Phase

The Blockchain Ledger, having informed the interested parties of the decided parameters and costs of a shipment, marks a transaction as "approved" or "denied" and sends the information to the Shipping Layer. Should the design of the shipment be approved, the Shipping Service repeats the functionalities described in the Design Phase, to obtain information on the containers, optimally pack them with the products of the transactions and calculate the best route to follow. Additionally, the Shipping Service examines the product constraints to be applied (e.g. temperature, humidity, light, acceleration) and communicates with the IoT Layer in order to obtain a unique API Key for each container, as well as to set up the tracking devices so that they monitor the conditions of the containers.

At this phase, the orders are marked as "In Transit" and the PI Model containing all the relevant information on decisions and settings is returned, as a digital Bill of Lading, to the Blockchain Ledger.



Initialization

Figure 3.4: Shipment initialisation

3.2.4 Iterative Phase

The processes and decisions described are repeated, in an iterative fashion, every time that a transaction reaches a PI Node. Upon arrival at a PI Node, the Shipping Layer examines the final Destination of a transaction: should this PI Node be its Destination, it marks the Order as "Delivered" and propagates the information onto the Blockchain Ledger. If this is an intermediate node and the transferred containers are almost full (i.e. no more products can be loaded), it communicates with the Routing Layer to obtain the optimal route from that PI Node to the Destination and, if required, reroutes the shipment.

Should the shipment be able to accommodate more transactions (i.e. from orders whose Origin is the current PI Node), the Shipping Layer repeats the Design and Initialization Phases in order to add the new orders into the In-Transit Shipment. In cases of Switch PI-Nodes where the containers shipped are to be packed to a different mover (i.e. from trucks to a ship), the Encapsulation Layer is called twice in order to optimally distribute the containers onto the new PI Mover.



Arrival at PI Node

Figure 3.5: iterative protocol upon PI container arrival at PI Node

3.2.5 Real Time Updates and Decisions

With IoT trackers in place, the Shipping Service requests real-time updates regarding the position of a shipment as well as the conditions of its containers. All measurements obtained are examined, validated against the product and shipment constraints placed and stored in the PI Model. Upon request by the Blockchain Ledger, the gathered measurements are sent for further examination and validation to the interested parties. If, at any point, a constraint is found to be violated (e.g. the

temperature of a container exceeds the permitted temperature for the products), the Shipping Layer communicates with the Blockchain Ledger to obtain the Destination where the shipment is to be disposed. Afterwards, obtaining the optimal route from the Routing Layer towards the new Destination, the Shipping Layer marks the transaction as Compromised and reroutes it for disposal.



Real Time Updates

Figure 3.6: Response to real time information acquisition by IoT

3.3 Shipping Service Implementation

The phases, functionalities and actions described in the aforementioned sections, can be best described in an Input/output table, as illustrated in Table XX. Any or all of the actions described above can be repeated as many times as necessary, pending receival of events such as arrival at PI hub (recalculation of the routing might be needed) or impeachment of order (part of order might need to be rerouted for safe disposal) etc.

Service/Function	From Service	To Service	Input	Output	
CreateOrder	Logistics Web Layer	Shipping Layer	Order arriving into PI ecosystem	PI enabled Order (uses PI data model)	
GroupOrders	Logistics Web Layer	Shipping Layer	Collected PI Orders	Transactions of meaningful and cost-optimal PI Orders	
GetDisposalDestination	Logistics Web Layer	Shipping Layer	Compromised PI Order	New Destination to which the shipment is to be diverted	
InitializeloT	Shipping Layer	IoT Cloud Platform	Transaction with container Id's API Key for IoT plati generated by item connectivity encapsulation		
ConfigureIoTTrackers	Shipping Layer	IoT Cloud Platform	API Key for IoT platform connectivity and product/shipment constraints, defined as a PI Order	PI Order updated with the successful or unsuccessful installations of tracking devices	
ValidateDestination	Shipping Layer	Network	Destination, as sent by the Web Logistics Layer	PI Node	
GetContainers	Shipping Layer	Network	PI Node and constraints (i.e. refrigerated containers required)	List of available containers that fulfil product contraints	
RouteOrder	Shipping Layer	Routing	Order with valid origin & destination	Order with optimized routing information using established links/corridors	
ExecuteItemEncapsulation	Shipping Layer	Encapsulation	PI transactions & products	Updated PI Order & PI container ids associated with products	
GenerateShippingInstructions	Logistics Web Layer	Shipping Layer	Orders that have finished initial composition/decomposition	Orders with corresponding shipping documents	
ApproveOrder	Logistics Web Layer	Shipping Layer	PI Order that has been approved by the stakeholders	None	
GetloTData	Shipping Layer	IoT Cloud Platform	Container ID and API Key	Tracking Data (i.e. GPS coordinates of each container and the condition of the alarms configured)	

Table 3.1: Shipping	Service fu	inctionality	Inputs/	outputs

3.4 Service sample application and design guidelines

3.4.1 Generic Shipping Service implementation sequence

To make the previously expressed functions clearer, an example from a generic implementation is described:

- 1. A stakeholder expresses the wish to ship good through the PI
- 2. The order and the relevant good are propagated through the Web Logistics Layer to the Shipping Layer
- 3. The Shipping Layer contacts the Networking Layer to obtain information on the order's Destination PI Node. The Destination's PI Node ID, Name, Longitude, Latitude, Functions and more information are retrieved.
- 4. The Shipping Layer contacts, once again, the Networking Layer in order to get information on the available Containers at the Origin.
- 5. Having a list of Products to ship, as well as the available containers at the Origin, the Shipping Layer instructs the Encapsulation Layer to calculate the optimal theoretical encapsulation. The encapsulated picking lists is returned to the Shipping Layer which, in turn, uses it with the Routing Layer which calculates the optimal route dependent on considerations expressed by the shipping party (e.g. CO2 emissions, fastest route, etc.) by utilizing data made available by the Network Layer. Calculation of transportation costs is also made here
- 6. After the routing is complete, the decisions made and associated costs of all services are returned, by the Shipping Layer, to the Web Logistics Layer for Shipper Approval
- 7. Either on regular time intervals or in an ad-hoc fashion for expedited deliveries, the Shipping Layer gathers all approved PI Orders and groups them in Transactions based on their Destinations, intermediate Nodes, product constraints (i.e. temperature, light, humidity etc.) and shipping constraints (i.e. time of delivery etc.). These Transactions will ensure that the goods will be shipped in a cost effective way, utilizing the minimum number of containers and movers required, the optimum configuration of IoT Trackers inside the containers as well as the optimal routes for the final delivery of all Transactions throughout the network.
- 8. After the PI Orders have been grouped into Transactions, the steps mentioned during the Design phase are repeated: the Networking Service is contacted in order to obtain the available required Containers on the Origin, the Encapsulation Layer distributes the goods into these containers and the Routing Service calculates the final, optimal route for the Transactions.
- 9. After the Shipping Instructions have been decided upon and established, the Shipping Layer obtains a unique API Key, from the IoT Service, with which it will configure and, subsequently, track the position and conditions of each container throughout its shipment.
- 10. All decisions and state of the IoT Tracking Devices are sent back to the Web Logistics Service and the Transaction is marked as "In Transit". The shipment of the goods can now take place.
- 11. While in transit, the Shipping Services get updates from the onboard IoT sensors, which are used to check against conditions as expressed in the smart contracts
 - i. In case of Impeachment (contractual obligations not met either by cost, time or product health) the Smart Contract validations will fail and all interested parties will be notified. In the case that the goods being transported require special handling, the Shipping Services reinitialize the **Routing** functions to ensure safe transport and disposal to the nearest appropriate facility

- ii. **In case of route unavailability** the Shipping Services are notified and reinitialize the routing functions to divert the shipment through an alternative route
- iii. In case of successful transit the goods arrive at the next PI hub, where the shipping service is notified (via uploaded proof of delivery and geofencing information coordinated with GPS location updates) and the in transit status is suspended.
- 12. After the goods arrive at the PI node, the iterative phase is executed (1...N times):
 - a. Shipping services trigger PI Container re-encapsulation. Stuffing/Unstuffing of boxes occurs (if needed) in order to optimize loads in transport means. Constraints are taken into account for this process.
 - b. Shipping services also trigger route checks for the next step in the end-to-end trip, and recalculate the route if needed.
 - c. Once all steps are concluded, the physical layer performs the necessary loading/unloading functions and notifies the Shipping Services once everything is done
 - d. The shipping services generate the next set of Shipping Manifests required, and mark the order as ready to be shipped
 - e. If order arrives at PI node N (final node in the trip), the products are unbundled and stored awaiting for pickup or last mile transport.

A high level view of the state management of the PI order can be expressed as a state diagram, found below.



Figure 3.7: States managed by the Shipping Protocol

3.4.2 Shipping Service in the PoC

Along with the description of the Shipping Services that is offered in the document, a first version of a generic software implementation has been deployed in the PoC environment. In this version, API endpoints are available for the initial steps of an order. An order can be passed & stored (in memory) in the Shipping Service software from the Web Logistics Layer and the flow of the PI can be initialized

by marking the order (containing products) ready for Item encapsulation. The order is then passed to the encapsulation service, after obtaining a list of available PI Containers from the Networking Layer.

After the successful Encapsulation of the goods into their respective containers, an API Call towards the Routing Layer is made, to obtain the optimal routes for the containers. Finally, the IoT Service is contacted, creating a Shipment for each container, obtaining the API Key of the installed trackers and configuring the trackers based on the product requirements. A scheduler, running at regular time intervals during the transport of a Transaction until its final Delivery Node, polls Tracker Data from the installed sensors and stores their information in memory – available via a Web Interface for the Web Logistics Layer.

Other functions, such as the shipping document generation have been designed but not fully implemented yet. The Shipping Service, however, updates the PI Order at all times and sends it (while making it available via Web Interfaces as well) to the Web Logistics Layer as a digital Bill of Lading.

As various functions are not in the scope of the shipping services and its software implementation is heavily reliant on availability and functioning prototypes of other layers, for this first basic version most of the functions are mocked-up. As such, the WP2 simulation toolset will utilize a mix of Shipping Service software functions and internal functions simulating the end-goal functionality. In future versions, a more complete and concrete set of services will be developed, with the end goal being a complete implementation of most, if not all, of the functions described above, whether they are part of the shipping service implementation or if they need to pass through the shipping service.

3.4.3 Design guidelines

In this section the end-to-end processes and functionalities of the Shipping Service are examined, in the context of each Living Lab.

3.4.3.1 PI Hub

In the case of the PI Hub, the Shipping Service is mainly responsible for the continuous monitoring of the mode specific traffic systems such as the Rail Traffic System for the PoA. It configures the IoT Devices within PI Containers and exposes all relevant information, via REST APIs, to the rest of the querying services.

Being able to distinguish between containers, via unique API Keys assigned by the IoT Service to each PI Container, the Shipping Service continuously updates the position and status of the shipments, exposing this information to the Web Logistics Layer, that in turn can use it to recalculate the ETAs, divert shipments in case of heavy traffic, prioritize wagons based on their expected arrival or departure times.

Whenever an order changes modes of transport (i.e. is loaded or unloaded onto a ship or a train), the Shipping Service calls the Encapsulation Layer that encapsulates the orders into PI Containers and/or onto PI-Movers, thus providing a pre-calculated picking list on each node. With such a list available, the Shipping Service manages the states of the orders (whether they're in transit, being loaded/unloaded, delayed), obtains their next PI Node via integration with the Routing Service and exposes this information to the Web Logistics Layer for further actions and decisions.

3.4.3.2 PI Corridor

The Shipping Service design for PI Corridors is focusing on materializing a Tracking Service, with the implementation and installation of IoT Devices (Smart Trackers or Smart Routers) on real-life containers transporting PGBS goods. The Shipping Service, in this Living Lab, translates the Smart

Contracts obtained by the Blockchain Ledger into IoT meta-data, configures the sensors in each PI Container and, using the SLAs imposed by the Ledger, monitors and validates their data throughout a shipment's lifecycle.

Should an alarm go off (i.e. the SLA is breached due to, for example, the PI Container encountering a bump along the way or delaying in a congested route), it transmits the event to the Blockchain Ledger that, in turn, will evaluate the situation and request a re-route, from the Routing Service if needed. Additionally, in locations where the mode of transport is changed, the Shipping Service contacts the Encapsulation Layer, that, in turn, distributes the goods into PI Containers and the latter into PI Movers, as needed.

Running as the overall orchestrator, the Shipping Service collects all data regarding the picking-lists (from the Encapsulation Layer), their routes (from the Routing Layer) and their real-time statuses (onboard readings and position information) and maps it into a comprehensive PI Model available for any other querying service.

3.4.3.3 eCommerce

The main function of the Shipping Service in the case of eCommerce is to track and manage orders that traverse its PI Network. In further detail, the Shipping Service updates the state of orders of goods, providing a real-time overview of the movement of these goods. By utilizing IoT Data from the Cloud Platform it exposes the measurements to the rest of the services (such as the Routing or the Web Logistics) enabling them to re-act or act pro-actively to conditions such as congestions, allowing for a re-route of trucks to take place, thus minimizing the lead times.

Keeping track of the product requirements shipped, the Shipping Service configures the IoT Devices in trucks, monitoring, collecting and exposing data such as their temperatures, shocks, humidity levels, illumination levels, acceleration and more. By exposing this data to the Web Logistics service, the Shipping Service establishes a thorough validation of the Quality of Service with regards to the shipment of a product. Finally, in order to allow for the better balancing of inventories, the Shipping Service exposes an order creation function towards the Network Service that might be used from the latter whenever goods need to be transferred from one warehouse to a different one, thus minimizing stock-outs which is one of the main goals of this particular use case.

4 ENCAPSULATION SERVICE

The encapsulation service is responsible for the optimal loading of cargo into PI containers. By registering available containers to be used for loading (populated by the Network and/or Physical layers), and by parsing the products/containers contained in an order, the functions of the encapsulation will achieve efficient assignments of cargo. The constraints, both standard and optional, are used to further limit which containers can be used for loading to satisfy the conditions needed by the products being moved, while at the same time adding further complexity to an already complex problem. By reducing the encapsulation algorithmic problem into a bin packing problem, extensive existing research and algorithms can be used to greatly improve optimization factors across end-to-end PI enabled shipments. The variety of these solutions to cover extra logic & constraints on a case by case basis. Along with physical interoperability and additional research being currently made (such as the GS1 standardization effort for smart containers) the resulting processes will allow for a greater degree of automation, resulting in increased load sharing, reduced co2 emissions in collaborative logistics. The first version of the encapsulation service deployed in the PoC environment, offers a software solution to the 3-dimensional bin packing problem addressing the appropriate constraints.

4.1 Encapsulation Service Considerations Overview

Following the original description of the encapsulation services in D2.4, this document aims to further specify the functions in a more concrete and technical manner. The encapsulation services, as described by the OLI/NOLI models, play a key role in the concept of the Physical Internet. The functions contained therein are responsible for logically converting the original packing list & relevant products stemming from legacy ERP or other non-PI systems into instructions detailing how these products will be stuffed into secure PI-containers. The resulting packing list will group the container Ids to be passed back to the invoking service (shipping services), where the data returned will be used to establish connectivity with the IoT equipment which are on board the containers. The functions mentioned are closely linked to the physical layer, as they depend on physical equipment and actions in order to stuff/unstuff products to PI containers. This connectivity will ultimately be used to provide initial status, composition states of the containers as well as container contents to other layers interested in the aforementioned information.

Additionally, the encapsulation services & the corresponding physical loading of products into containers are the first step in enabling the Track & Trace capabilities of the PI paradigm. The IoT events that are responsible for monitoring and tracking shipments are made available by initially providing the IoT cloud platform the container IDs of the PI containers being used, as expressed by the picking list calculations.

As it is apparent, the algorithmic approach outlined in the next section is of critical importance for efficiency and reliability of the encapsulation functions and in turn, the long-term goals of the project. Emission reduction, load sharing and collaborative logistics are clearly affected by the outcomes and performance of these services.

4.1.1 PI-containers

Encapsulation is the activity of stuffing goods into smart, world-standard, modular and designed-forlogistics containers. Those containers, which we refer as PI-containers simplify and automate most of the processes in the whole supply chain such as, transshipment, cross-docking, sorting, and consolidation of unit loads. They also ensure the privacy and safety of goods.

Several publications have introduced generic specifications concerning the functionality and dimensionality of PI-containers, such as in Montreuil (2009, 2013). Concerning functionality PI containers make all the interconnected-logistics operations. They are modular and allow their composition into larger containers and the decomposition into smaller making easier the load and unload operations. Concerning intelligence, they are smart, uniquely identifiable, exploiting the Internet-of-Things technologies for traceability, security, and safety purposes.



Figure 4.1: PI containers

Product encapsulation can be applied on several layers. In the Physical internet usually, we have four encapsulation layers. These include: First, the level of wrapping of the products as we find them on the shelves of retail stores (Packaging Layer). Second, the level of handling packages (Handling containers) which in the case of physical internet means the enclosing of products in light boxes safety, easy to handle for loading-unloading, traceability and best use of the space inside the transport containers which constitute the third encapsulation layer. Finally, the ability of T-Containers to combine with each other and make larger composite containers makes easer the last mile transport as they can be easily adapted to the dimensions of trucks (fourth layer). Figure 2 depicts the flow of items in the encapsulation service.



Figure 4.2: PI encapsulation layers

As we note in Figure 4.2 the existence of the four encapsulation layers does not mean that all products should go through all levels of encapsulation. For example, after the packaging layer, the products can be placed in the transport containers. In the following we present a brief description of the encapsulation layers.

4.1.2 Encapsulations layers.

4.1.2.1 Encapsulation layer 1: goods packaging (P-containers)

Packaging containers are light weight boxes that contain the goods' packages as they are displayed in retail stores. Usually these are encapsulated inside the containers of the next layer (handling

© ICONET, 2020

containers). Concerning their size, they follow the same scaling factors as the containers in the next layers. Certainly, they should be smaller than the smallest H-container $(1.2x1.2x1.2 \text{ m}^3)$

4.1.2.2 Encapsulation layer 2: Handling containers (H-containers)

Packaging containers, such as cardboard, totes and boxes are encapsulated inside handling units. From a logistics perspective, the cubic format of H-containers makes them easier to handle than odd-shaped loads, such as on pallets.

Handling containers in PI are upgraded with the generic specifications of PI-containers. H-containers are lighter, and smaller than the containers of the next level (T-containers) where they are encapsulated for easy to transport and safety reasons as it is depicted in **Figure 4.3**.



Figure 4.3: A T-container filled with H-containers

Several models of H-containers have been proposed in the literature. In **Figure 4.4** we give an example of a prototype H-containers proposed in the Modulushca project [1].



Figure 4.4: H-container proposed in the Modulushca project

The size of H-containers depends on the size of the T-containers in which they are stored with the largest H-container to fit inside the smallest T containers. Their size is a modular multiple of a basic H-unit container (H_b) defined by:

$$H_{container} = f \cdot H_b \ H_b = \frac{T^{\text{int}} - 2 \cdot thickness}{10}.$$

Then, where *f* is a factor with an integer value $f \in \{1,2,3,4,5,10\}$, T_{int} is the interior dimensions of the T-container and s is a small gap between interior side of the T-container and the encapsulated H-containers.

4.1.2.3 Encapsulation layer 3: Transport Containers (T-containers)

Transport containers are functionally like the current shipping containers, yet with the upgrading generic specifications of PI-containers. T-containers are world-standard, modular, smart, eco-friendly and designed for easing interconnected logistics. They can sustain hard weather conditions and tough seas. Like the current shipping containers, they are stackable on several levels. They are to be stackable at least as many levels as the current shipping containers. From a dimensional modularity perspective, their external height and width are to be 1,2m or 2,4m while their external length is to

be 12m, 6m, 4,8m, 3,6m, 2,4m or 1,2m. Their size is a modular multiple of a basic T-unit (T_b) defined by:

$$T_{container} = f \cdot T_b.$$

where f is an integer parameter which takes values {1, 2, 3, 4, 5, 10} and Tb is a basic T-container.

Several types of T-containers were proposed which all are modular combinations of a basic container. This basic container was defined as a cube with edge of size 1.2m. In this way we can define several types of T-containers with an identification reflecting their dimensions. In **Figure 4.5** we present T-containers of several sizes. In the following we use the notation T.w.h.d to denote a PI-container with external dimensions: width=w*1.2m, height=h*1.2m and depth=d*1.2m. Thus a T.2.2.2 container has external dimensions width=2.4m height=2.4m and depth=2.4m, and a T.2.2.10 container has width=2.4m height=2.4m and depth=12m.

The sizes of T containers above, refer to their external dimensions. For the internal dimensions, a space corresponding to the thickness of the container must be considered. We should node that these dimensions of T-containers are indicative, and their final values need further investigation.

4.1.2.4 Encapsulation layer 4: Transportation (carriers)

For sea transport PI-containers can composed to T.2.2.10 containers whose size is approximately the same as the existing ones. For the last mile transportation in land goods are loaded into carriers. Such carriers include several types of vehicles such as road-based trucks, semi-trailers, delivery vans. Given that the PI-containers can be combined and create complex containers of different sizes, the means of transport mentioned above can be used. This is a positive element of PI- containers as opposed to current constant size containers.

Having T-containers of several sizes makes easier their shipping on several transport-modes mainly in land transportation.





Figure 4.5: Several sizes of T-containers proposed in the literature

4.2 Encapsulation Services

The encapsulation services have four clear functions.

- <u>Item Encapsulation into H-containers</u>: This function refers to the packing of products into H-containers, depending on product/order requirements such as the list of items with their dimensions and weights, their location of origin and destination, the type of the goods (temperature, humidity, dangerous freight, etc.) and outputs a corresponding packing list, containing the container ids to be used to establish communication with the IoT cloud in order to receive events concerning aforementioned containers.
- <u>H-container Encapsulation into T-containers</u>: This function can be triggered each time a shipment arrives at a PI node, in order to further compose/decompose containers into other containers, depending on the parameters expressed initially in the order & various optimization parameters.
- <u>Shared T-container Encapsulation</u>: This function can be triggered each time a shipment arrives at a PI node, to compose/decompose smaller orders of T-containers into other containers, (ship size containers (T2.2.10) or in the case of land transport (train, truck, barges) to adapt the encapsulation in the size of the transport mode. Again, the procedure depends on the parameters expressed initially in the order & various optimization parameters.
- <u>Initial Picking List (s) Generation</u>: During the cost calculation phase, a first step encapsulation
 is performed (without the physical implementation), that outputs an initial Packing List. This
 list is then passed on to other services and is used to calculate costs and plan the initial
 routing. In order for the encapsulation functions to achieve the optimal result, the
 corresponding services need to be made aware as for what containers are currently
 available in the node that will execute the encapsulation (either item or container). In the
 following we will assume that enough PI-containers are available and the Picking List
 Generation process involves the selection of the containers which minimize the number of
 containers required to encapsulate the products and maximize the space used. This function
 will be used to calculate the delivery with minimum cost.

4.2.1 Constraints

As mentioned in D2.3, one of the major areas of focus in the encapsulation services is fulfilling various constraints deriving from the original order. In this version of the deliverable, an attempt is made to list all the constraints that are taken under consideration when designing the encapsulation service and it's functions, as well as how they interact with each other.
<u>**Dimensions</u>**: The standard constraints that limit the algorithmic output (as described in the following section) will always be dimensions. As expected, a container can be filled only with products with dimensions that allow that product to be physically stuffed in the container.</u>

<u>**Orientation**</u>: A secondary constraint, very closely attached to the dimension's constraints, is an orientation constraint. If a product is rotation-free then there are 6 possible rotations to try before inserting the object into a PI-container. However, some large electronic devices used to have warnings that they should be moved in an upright position. For these items only two rotations can be tested in order to fit then inside a container.

If xi, Xi are the dimensions of the items and the containers respectively then if the item is rotationfree, it fits in the container if $\forall x_i, X_j, i, j=1,2,3$ hold that $x_i \leq X_j$.

If the item is limited to only two rotations then it fits in the container if x_1 , $x_3 \le X_1$ or x_1 , $x_3 \le X_3$.

Weight: Weight constraints and overall load bearing in a PI container is another important constraint that the encapsulation processes should take into account. Exceeding weight capacities or mismanaging load bearing could result in damages of cargo.

Spacing between goods: Load bearing is also affected by product constraints, such as required spacing between cargo units (in the example of fresh produce, specific placements allow for needed air flows) inside a container.

Type of Goods: On a similar manner, some products might not be able to be grouped together with others, such as food with chemicals. Humidity, shock, vibration and temperature are all product constraints that might affect primary constraints in one way or another. In summary, the various constraints which will be taken into consideration are the following:

- Dimensions
- Weight
- Load Bearing (weight, spacing in between)
- Type of Goods (Temperature, Humidity Dangerous cargo)
- Grouping / Placement

While some of the constraints that were described are always taken into account when performing encapsulation operations, some others are entirely optional and dependent on specific use cases and modes of transport. Dimensional and Weight constraints will have to be always taken into account in any operation, while Load Bearing for example is negligible in e.g. train transports. In a similar vein, temperature, humidity, shock, grouping and cross contamination constraints are exclusively dependent on the types of products being moved.

Apart from the item encapsulation, and the constraints that were already mentioned, the encapsulation service is also responsible for stacking containers within other containers. This process also inherits some of the constraints deriving from the products that are within the containers to be encapsulated, but it also needs to take into account routing and delivery time constraints, to perform the optimal encapsulation guaranteeing on-time and as-ordered delivery.

4.2.2 Encapsulation and Consolidation Protocols

The encapsulation protocol specifies how products are assigned to a PI-container or a set of PI-containers, how to decide on the size(s) of PI-container(s) to be used for each shipping group, and how

to decide on the loading sequence and pattern of goods within each PI-container. Given a set of allowed container sizes, the selection of specific containers for every order (origin, destination, date) here aims first to minimise the number of containers and second to maximize space utilization.

The consolidation is a clustering activity of the orders to be shipped within the same time period and having a common destination, i.e. a warehouse. In this case we don't have unloading and loading of PI-container contents in intermediate hubs. The container consolidation protocol aims to fully load the selected transportation services. The protocol optimizes the consolidation of containers given by next destination or the subsequent destinations in order to avoid as much as possible PI-container unloading, and loading operations. Thus the consolidation of containers is closely linked with the routing service.

Given that the delivery time of the products is a key factor in the smooth conduct of transport, the consolidation protocol must give the highest priority so that they will be immediately transported. For the optimization of loading a bin-packing algorithms used that will be discussed in the next section.

4.3 Algorithmic Approach

4.3.1 D3 bin packing algorithm

In this section we will deal with the problem of bin packing. We will describe the algorithms we developed and the experimental results of packing T-containers with H-containers, in the context of the work package 2.3. In the **bin packing problem**, items, usually of cuboids' shape of different volumes must be packed into a finite number of bins or containers, not necessarily of the same volume in such a way to minimize the number of bins used. In our experimentation we used simulated data for fitting a random number of H-containers with randomly chosen dimensions but compatible with the sizes of Pl-containers as were described in the previous section.

Formally the bin packing problem is defined as follows:

Given n items with sizes (wi,hi,di) to be packed in an number of N bins of size (W, H,D) each, the aim is to use the minimum number of containers and to maximize the space used. By (wi,hi,di) and (W,H,D) we denote the width, height and depth of an item or a bins respectively.

In a strict mathematical formulation the problem is defined as follows. Consider two variables X, Y defined by:

$$X_{ij} = \begin{cases} 1 & \text{item i, ispacked in bin } j \\ 0 & \text{otherwise} \end{cases} \text{ and } Y_j = \begin{cases} 1 & \text{bin j is used} \\ 0 & \text{otherwise} \end{cases}$$

Then the bin packing problem is defined as the integer optimization problem

minimize
$$\sum_{j=1}^{N} Y_j$$
 subjected to the following conditions:
 $\sum_{j=1}^{N} X_{ij} = 1$ for $i = 1,...,n$

```
\sum_{j=1}^{N} vol_i \cdot X_{ij} \le Vol \cdot Y_j \quad \text{for } i = 1,..., NX_{ij} \le Y_j \quad \text{for } i = 1,..., n, \ j = 1,..., NX_{ij} \in \{0,1\} \quad \text{for } i = 1,..., n, \ j = 1,..., NY_i \in \{0,1\} \quad \text{for } j = 1,..., N
```

The first condition forces the placement of each item into one bin. The second condition represent the upper limit on the bin's contents, as well as the fact that items cannot be packed in a bin that is not in use. The third condition indicates that we cannot place items, (Xij=0), inside a bin which is not used (Y_j =0).

In computational complexity theory, this is a combinatorial NP-hard problem [3]. NP-hardness (nondeterministic polynomial-time hardness) in computational complexity theory, is the property of a class of problems that are informally "at least as hard as the hardest problems in NP"¹. This means that there is no a computationally feasible optimal solution for the problem.

Despite the fact that the bin packing problem has an NP-hard complexity, optimal solutions to a very large number of instances of the problem can be produced by several heuristic algorithms. Such algorithms like First-Fit or Best fit algorithms require $\Theta(n \log n)$ computational time, where *n* is the number of items to be packed.

Loading conditions subject to several restrictions, such as stacking constraints, orientations, horizontal only rotations, conditions for special cargo such as temperature, humidity, dangerous cargo etc. In our implementation to fit them in the bin we rotate the items in 6 possible orientations. Restrictions concerning the temperature -humidity of products are dealt in a previous step separating the products into different groups and applying separate packing algorithms for them. In the following subsections we describe the algorithms we developed in the context of the project. The FFD (First Fit Decreasing) and BFS (Best Fit Search) are two well-known heuristic algorithms developed here for the D3 bin packing problem while the Modified FFD algorithm is an innovative algorithm that significantly improves the performance of the FFD as is presumed from our experimental results.

4.3.1.1 First Fit Decreasing Algorithm

```
for each item to Pack
    insert the item in the first available bin if it is fitted
    else:
    insert the item in a new bin
```

An item is fitted in a bin if does not overlap with other items inside the bin. For this purpose, depending on the orientation constraint, we examine either 6 or 2 rotation of the item to check if it is fitted in a bin.

The packing of a bin starts from the back-bottom-left corner of the container, point (0,0,0). The next item is positioned in the direction of w (width) at point A, as it is shown in figure 5. If there is no enough

© ICONET, 2020

¹<u>https://en.wikipedia.org/wiki/NP-hardness</u>

space in this direction we position the item in the direction of h (height, point B) otherwise we insert it in the direction of d, (depth).

Before packing the items are sorted in decreasing order of their volume, a process that improves the efficiency of the algorithm.



Figure 5: Packing items inside a bin

4.3.1.2 Best Fit algorithm

```
for each item to Pack
```

```
insert the item in the best available bin if it is fitted else:
insert the item in a new bin
```

Best Fit packs an item into the bin with the least available capacity. If there is no space enough to fit the new item, then the item is inserted into a new bin.

4.3.1.3 Modified FFD algorithm

Our experiments from both, FFD and BFS algorithms showed that if we have a relatively large number of items to pack, both algorithms achieve almost optimal solution. This makes sense since both algorithms keep several bins open simultaneously so most of them are fully packed. This is certainly a satisfactory theoretical result, but in practice this means that various items are being stored in different containers, which greatly increases the cost of transport as many containers should be loaded/uploaded to serve an order. However, this is not the case when the items are just a few enough to fill one container. In this case the performance of both algorithms can be very poor, achieving between a 50-60% usage of the containers.

For this case we ran a number of experiments each with a number of items that fit exactly one container and for which both algorithms, First Fit and Best Fit, fail. This is an interesting problem which currently, although done by specialized and experienced staff, is not at all an obvious task. The possibility of such a quick solution serves cases that require the issuance of an invoice for the transport of products. For this problem we have implemented an improved version of the First Fit Decreasing algorithm which tests more and different sequences of the items to be packed. As we already mention we rotate an item in order to test if it is fitted inside a bin or not. We don't test all possible rotations (6 or 2) for each item since this lead to exponential complexity. The algorithm insert an item in a bin when find a rotation that fits the item. However, for the first items to pack, because there is still more space available in the bins the items do not rotate and empty space may be created inside the container. The same situation is created with the latest items to pack.

To overcome this problem we swap the items locally to change their order of packing. For a given set of n items, this implies 2n swaps between neighboring elements. Then the algorithm is applied for each sequences on input data, 2n times or until items were fitted into the smallest number of bins (which is 1 bin).

This refined version of FFD was tested on several examples of random packing simulations with randomly generated dimensions but in accordance to the dimensions described in section 2. From these data a set of 110 different sequences of items that fit exactly inside one bin and for which both algorithms, FFD and BFD, fail was created. Out of the 110 examples, the proposed algorithm found the optimal solution in 68 cases.

In our evaluation process we have used two measures for the used space in the containers defined by:

 $micro Average Usage = \frac{Total capacity of input items}{Total capacity of used containers} and$

macro Average Usage = $\frac{\sum_{j=1}^{N} U_j}{N}$, where N is the total number of bins used and U_j is the proportion of the space used per bin:

 $U_{j} = \frac{\text{Total capacity of input items in the bin}}{\text{Total capacity of bin}}$

In Table 1 we give the input data, the available containers and the corresponding results.

From our experiments it is evident that the algorithms could substantially improve if the input data are given in the right order [5]. In this vein we are currently investigated techniques from AI, using deep reinforcement learning for the prediction of that order of the input data that will give us the best performance out of the FFD algorithm. However from our experiments so far the results weren't improved. A much higher train set definitely should be used with and a much powerful computational effort.

4.4 Service sample application and design guidelines

The first version of the encapsulation service deployed in the PoC environment, provides functionality for bin registration and bin packing. The bin registration function allows other services (usually from network or physical layers) to register PI containers as available for loading. Following the initial bin registration, functionality is provided in order to register and encapsulate products optimally into these containers. This process can be then reused as many times as necessary in order to further encapsulate containers within containers. As this is the first version of the encapsulation service, the constraints taken into account when performing encapsulation operations are limited to dimension and weight constraints. In following versions, additional constraint logic will be added, working towards integrating all the constraints as listed in previous sections.

4.4.1 PI Hub operations and encapsulation optimisation

As part of LL1-PoA (Port of Antwerp), an optimization service is being developed with the objective of optimizing train wagon cargo loading. The specific use case problem is also approached as a bin packing problem and more specifically, as a single bin packing problem, as the goal is to optimally fill each wagon on the train. As expected, this implementation also takes into account the specific

constraints of the particular problem. The major constraints the optimization service will focus on are weight limits of the wagons, overall length of the train, as well as priority of the wagons for loading. This optimization service is planned to be coupled with the Rail Traffic System, an overall software/hardware solution currently being developed by PoA to better handle traffic by tracking trains by various hardware means and intelligently assigning loading/unloading station slots to incoming/outgoing trains. A detailed description of the combined operations and encapsulation optimisation provided by ICONET PI Optimisation Service can be determined in Deliverable D2.13 & D2.14.

4.4.2 Design guidelines

In this section the end-to-end processes and functionalities of the Encapsulation Service are examined, in the context of each Living Lab.

4.4.2.1 PI Hub

In the case of PI Hubs such as the Port of Antwerp, the Encapsulation Service is mainly responsible for loading/unloading train wagons based on the bin packing algorithm developed for this purpose.

The encapsulation layer waits for a call from the Shipping Service to load or unload a list of Plcontainers. The call will contain all the necessary parameters and restrictions necessary for the encapsulation. These parameters include: the Identifier of the wagon, its dimensions and information concerning the items to pack. Such information is: the location of origin and destination, the sender, receiver, the transport mode, the type of goods, the identifier of the items to pack their dimensions and their weight. Two types of restriction are applied: the maximum weight capacity of the wagon, and the rotation type of the items. Two types of rotations are applied: general-rotations which apply all possible rotations (6) and, restricted-rotation when should keep the upwards dimension unchanged (2 rotations). The encapsulation returns to the shipping service the container's identifier, its total weight, all the parameters of the encapsulated items, with their position inside the wagon.

4.4.2.1 PI Corridor

In the case of a multimodal corridor the encapsulation layer will select the optimal number of transport containers to pack the items, those which maximize the utilized space and minimize the number of containers. The algorithm with the parameters and restrictions are the same as in the case of PI Hubs. At the locations where the mode of transport is changed, the Shipping Service contacts the Encapsulation Layer, to unload the items from one mode and load them in transport containers of the other mode. This operation may mean removing all items from a container and reloading them on a train's wagons as in a PI Hub and vice versa. In all the cases the encapsulation service returns to the shipping service all the details of the encapsulation.

4.4.2.2 eCommerce

The main function of the Encapsulation Service in the case of eCommerce is the loading/unloading of trucks or trains that traverse the PI Network. In this case the encapsulation takes as input the dimensions of the platform of the truck or in the case of train the dimensions of the wagon and their maximum weight capacity. Finally, the loading algorithm takes into account the next and subsequent hubs between the origin and the destination. The output is similar as in the case of a PI Hub.

5 NETWORKING SERVICE

The Networking Service is the process that determines the infrastructure and services available for navigating the cargo towards its destination (via the PI network). It acts in essence as a digital twin of the PI enables transportation infrastructure, representing all its components, services, as well as their operational status. The Networking Service is therefore, both:

- and a short term/ operational process that needs to track the status of infrastructure and services in the network in live time, as this has a direct impact on the operational cost of the facilities, lanes and corridors that will be utilised.
- a long term/. process that needs to consider the longer-term state of the network facilities and other financial and non-financial parameters of the, facilities, lanes and corridors that will be utilised

Both long- and short-term planning for transport through PI will require collaboration with the infrastructure managers of the corridors followed in the planned path, and the intermediate nodes and facilities that will be utilised. The role of Networking Service is to establish (collect, analyse) all crucial information for the operation of the PI network and in association to PI infrastructure. Aside the network discovery functionality, the Networking Service need to be capable to package information when responding to requests in a robust and efficient manner. It is there a requirement to develop a packaging tool that will filters the network information relevant to a specific PI Order.

To facilitate the network discovery function, a PI network components data structure is proposed that covers all the information collected and analysed by the Networking Service. In order to enhance communication and decision making capabilities of other PI Services that depend on Networking Service information, a common data structure is proposed that has its foundation in the GPICS. All PI network components are associated to static and dynamic information, with dynamic information ranging over several sub-categories that are discussed in 2.3. The network identification covers nodes, links and services that operate in each, while additional categorisations are made for each physical infrastructure component with properties to be used for operational decisions.

The Networking Service is therefore responsible for collecting, storing and integrating all PI network information and communicating PI relevant network information in an interoperable and interconnected manner.

Chapter 5 aims to present the Networking Service protocols, and builds on the from Deliverables D2.3 and D2.4 that presented rawer versions of the service, focusing primarily on associations with the Digital Internet and the core components of its functionality. Section 5.1, further expands the discussion on Networking service design considerations, focusing on the generic representation of networks based on graph theory, and the PI network's data structure, including a reference on the GPICS that are used as its foundation. Section 5.2 focuses on the protocols of the PI Networking Service, covering:

- the Network discovery module
- the information packaging module
- the overall communication's with other PI services
- extra components that have been developed to enable the expansion of the PI applicability to more industries and sectors such as the eCommerce and urban logistics. Subsection 5.2.3

presents a cost minimisation problem for identifying the optimal allocation of fulfilment stores for eCommerce shopping orders.

Finally, the networking service representation of the TEN-T network is presented in Section 5.3.

5.1 Networking Service Considerations

5.1.1 Representation of logistical network features

Every complex network can be formally represented as a graph G = (N, L) consisting of a set of nodes N and a set of links L. Links can be directed or undirected (i.e. serving both directions) and are often associated with attributes. In transportation networks link attributes may include length, travel time or capacity of a link (Ahuja et al., 1993). The basic components that define networks are listed below:

- **Regions (or zones)** are areas of the graph that enable the grouping of homogeneous elements.
- Source and sink nodes are fictive points in a region from which regional flow originates or concludes.
- Feeder links (connectors) connect a source/ sink node to the rest of the network.
- Nodes resemble points of interest or intersections.
- Links (edges) are connections between two nodes.

When considering the Physical internet it becomes evident that there a more refined network representation is reqired. The ICONET Port of Antwerop (PoA) Living Lab can be used to illustrate this concept. In the representation of the PoA there are several interconnected nodes including:

- 26 container terminals are located in the port area (18 on the Right bank and 8 on the Left bank)
- two rail terminals with the major hinterland connections originating from the Right Bank)
- the adjucent road/ highway network
- burge (river boat) terminal at the Right bank

At the same time, in the context of Europe's TEN-T network, Antwerp is represented as a single node, interconnected with other north Europen cities. Beyond the terminal facilities, the PI hub representation is also required to model the end-to-end voyage of cargo within the hub. This is an essential requirement as in the more refined representation of a PI hub there are several sources of cost and uncertainty that require to be considered. For example:

- customs proceedures,
- localised capacity constraints for movements around the port,
- congestion at the entry/exit gates,
- errors in documentation.

It is therefore essential to introduce a formal representation of the network that covers the most impactful aspects of a ports (or any PI hubs) operations. The following subsections investigate the PI Network representation into further detail.

5.1.1.1 GPICS Aggregation level

Networks can be considered at different levels of abstraction, or in terms of layers. This helps with the definition of protocols that operate at the intra-layer and inter-layer level. For example nodes in lower tiers are connected to each other in the Physical Internet via π -hubs that form networks at higher tier levels. Nodes in lower tier networks can be considered to be connected directly to each other (within the same network), while PI-hubs form a connectionless layer (similar to the IP layer of the digital Internet). This connectionless architecture allows the Physical Internet network to be formed as a network of (transportation) networks.

5.1.1.2 PI as a Queue Network

In electrical and computer systems and networks as well as in business systems, architectures and processes are often modelled using concepts from queuing theory. In this report we consider models comprising queues servers and flows that can be used to represent key network and processing characteristics of the Physical Internet. Formal models of PI can be used for simulation and mathematical analysis of properties such as performance (throughput), delays and other PI parameters of interest.

A queue is formed when there is competition for limited resources. Physical Internet moves physical objects packaged in π -load units² (e.g. π -containers) along a path of nodes (' π -hubs') from a source to a destination.

The fact however that a π -type load unit and the objects it contains have to go through intermediate nodes towards the final destination, entails that queues of load units arriving to a π -hub will be formed, provided that the rate of arrivals of such load units is higher than the processing rate (throughput) of the hub.

So the simplest way to understand a π -hub as a queue is as per **Figure 5.1** The circles inside the open sided rectangle of **Figure 5.1**, are the π -'objects' (let's use this terminology from now on rather than for example terms such as ' π -containers', as there can be several types of π -specific load units). These π -objects are 'served' by the hub (the types of service/ π -related operations are discussed in the bibliography sources) and therefore spend some time in a queue and eventually exit the queue/hub towards the next hub ('next hop') or the final destination hub.



Figure 5.1: PI hub as a queuing system

In fact, the view of a π -hub as a single queue system is not very useful for any kind of detailed simulation or analysis. Different π -objects would be subjected to different operations within a hub and/or routed to different destinations. Each queue can for example, be dedicated to one type of processing, to one transport mode, or to one route. Thus, it is more useful to view a hub as consisting of multiple queues as per **Figure 5.2**. In that figure, a processing function routes each incoming π -object to the correct queue.

² A **unit load** combines individual items or items in shipping containers into single "**units**". We assume these are PI types of loading units such as π -containers.



Figure 5.2: A hub consisting of different queues

In fact however, a π -hub and its operation cannot be studied in isolation as it depends on the behaviour of the other π -hubs it directly connects to. This is explained in the next section.

5.1.1.3 π -hub forward and feedback flow dependencies

It is more useful to consider two adjacent ${}^{3}\pi$ -hubs as connected with circular transport flows. In this view, ' π -movers' (i.e. freight trains, trucks, liner ships,..) transport π -objects back and forth between the two hubs. Thus, a hub is both a sender and receiver node in the PI network, or from a graph theoretic perspective, there are loops in the PI network/graph formed between adjacent nodes.

From a performance (delays/throughput) perspective, the performance of a hub depends on the rate of arrival of shipments which in turn depends on the performance of adjacent (directly connected) hubs⁴.



Figure 5.3: The PI represented as a queue network

In **Figure 5.3** Hubs A and B, and Hubs B and C exhibit such interdependent (feedforward/feedbackward dependencies⁵). This does not necessarily mean that for example, π -objects leaving Hub B for Hub C will eventually re-appear as inputs to Hub B. It means that the performances of Hu B and Hub C (and also of Hub A and Hub B) are interdependent.

Figure 5.4 shows a similar situation to **Figure 5.3**, but with hubs operating multiple queues (although no feedback flows are shown).

 $^{^3}$ Two π -hubs are adjacent if there is a direct connection between them that do not involve another π -hub

 $^{^4}$ Assuming the transit time of $\pi\text{-objects}$ between the hubs to be approximately constant.

⁵ In an open feed forward queuing network, a job cannot appear in the same queue for more than one time. In an open feedback queuing network, after a job is served by a queue, it may re-enter the same queue



Figure 5.4: A PI queue network with 4 π -hubs

In Figure 5.4 transport of π -objects from Hub D to Hub C (and vice versa) is through 2 hops- Hubs A and B. Representing the PI as an open⁶ queue network with each π -hub as a node comprising multiple queues has several advantages in terms of the potential for formal analysis and (discrete or qualitative) simulation purposes.

Interesting properties of PI can be obtained analytically using mathematical properties of queuing networks. Questions such as average, minimum or maximum quantities of parameters such as the waiting time at queue in a π -hub or the number of π -objects present at anytime in a π -hub or the entire PI can be answered. Such answers can then be utilised by algorithms that route π -objects along the PI. Also, design of the PI network can be optimised by including for example temporary storage in suitable parts of the network (' π -storage') to smooth performance.

5.1.1.4 PI Link Weights

Cost, time, and reliability are key Link performance indicators from a shipper's perspective capturing the punctuality, time and cost for transit from the factory (or other production site) to the final customer. Performance of the corridor for different types of shipment (e.g. domestic versus international) need to be considered and different corridors compared on this basis.

The types of traffic along a corridor must also be considered. Although most corridors carry multimodal traffic, some are configured to carry specific types of traffic. A PI connected corridor is expected to become more multimodal as traffic enters and exits the corridor through different mode nodes (terminals).

Financial characteristics and performance indicators such as the ratio of the cost of logistics to the value of the delivered product, and the ratio of free on board (FOB) to cost, insurance, and freight (CIF) prices.

⁶ We call it an 'open' network because π -objects do not stay permanently in the PI but exit at the appropriate points (π -gateways, according to the PI nomenclature)

Non-financial Corridor Metrics to be considered include:

- The time taken to transit the whole corridor and each part of it for the type of goods considered (e.g. fast moving, perishable etc).
- The cost to importers or shippers to move cargo over the length of a corridor.
- The frequency of services and the expected wait time for the whole corridor and each of its components
- Reliability: The variation in time and cost for the whole corridor and each part of its components (reliability) could potentially be impacted by both quantitative and qualitative changes to the transport patterns.
- Security& Safety: The security of goods transported in the corridor and the safety of the people involved in that transport for the same reasons as the other metrics above.

Link weights can also be used to assimilate the effects of queues at nodes. For example, in modelling road networks, the amount of traffic that can transverse a traffic light intersection is measured through the notion of saturation flow. That is the number of vehicles that can exit the intersection in a hypothetical hour of green light. The same intersection can also be associated to a travel time weight, that represents the expected time loss at the traffic light. This concept is often used to model queues, and at the same time simplify the representation of the network.

5.1.1.5 Intramodality

Hubs connect multiple origins to multiple destinations. They may also perform additional functions to the cargo that passes through them. If incoming cargo is only transhipped, the hub can be considered to be a transit or relay hub. Intermodal terminal hubs are therefore the interface between the different transport modes and thus are key to access intermodal transport services and to ensure efficient and road-competitive intermodal supply chains throughout Europe.

Besides the pure transhipment of loading units from one transport mode to the other, intermodal terminals under PI have to perform several basic functions such as:

- Trans-shipment of loading units between different transport modes.
- Check in/out functions, such as check of import/export documents, the security and damages to loading units, handling of dangerous goods and respective documentation etc.
- Provision of transport means, such as rail engines and truck. Facilities such as cranes for loading and unloading, of cargo.

Facilities for internal transhipments and temporary storage of cargo.

In the Physical Internet, the Networking Layer comprises the interconnected infrastructure of processing, storage and transporting facilities (transport services, terminals, distribution centres, warehouses) through which the goods will be transported from their origins (manufacturing, distribution and other locations) towards their customer(s) locations.

Conceptually, the Physical Internet employs an architecture similar to the digital Internet, with smaller networks (similar to Internet's autonomous systems-AS) connecting to each other via gateway/routers (implemented by the π -hubs) and forwarding (physical) packets of cargo from origins to destinations.

Thus the Physical Internet protocols share many characteristics with their equivalent digital Internet protocols. For example, to route physical packets (π -packets) through the Physical Internet, router nodes must employ routing protocols similar to the Internal and Border gateway protocols.

When packets are routed inside an autonomous system, any routing protocol preferred by the administrator organisation can be employed of course. However, for packets that are routed outside the boundaries of the autonomous system, specific gateway protocols must be employed. These must be agreed between all the interconnected π -hubs. π -hubs must exchange routing information and synchronise with connected π -hubs.

5.1.1.6 Integrated representation

As discussed in the previous sections, the PI features and complexity dictate, a comprehensive, yet as simplified as possible network representation. This is best achieved through the consideration of appropriate weights, such as travel time, congestion, frequency and reliability of services for PI Links. Then a fictive source and sink node is assumed for every PI Hub, as illustrated in Figure 5.5. This pproach allow for the additional representation of within the PI hub operations or properties.



Figure 5.5: PI hub transhipment operations representation

In Figure 5.5 the transhipment operations links that connect every arrival mode (source nodes), to every departure mode (sink nodes) enable the assignment of weights to PI Hub operations. Transhipments can be associated to:

- Distances that require to be covered. This weight is more relevant to large PI hubs such as the PoA where transhipment legs are of considerable distance.
- Cost that can incorporate average handling cost
- Travel time that can incorporate average handling time and queues
- Capacity that represents the number of such transhipment PI hub infrastructure can undertake.

A similar network representation can be adopted for capturing PI hub specific costs and features, such as local congestion, queues at customs, various routes and their capacities, storage capability, cross-docking facilities.

5.1.2 Data Structure

With the network representation of the PI presented in the previous section including a variety of weights and operations representation, it requires to be associated to a similarly comprehensive data structure. This data structure builds on the data structure provided by the GPICs and described in Section 2.3 of this report. Following the graph theoretic approach to the representation of networks, the data structure requires to consider PI Nodes and PI Links and their properties. Additionally, as the

PI, approaches the transportation of cargo as data packets, it requires an additional element that is not present in digital networks, that of PI movers. PI movers are the rolling stock available in the Physical Internet on which PI containers, the PI cargo, are transported. Being an integral part of the PI, PI movers and their properties also require to be considered in a data structure definition.

Another perspective of the PI networks data structure is that of data detail. A network of nodes and links can be accurately represented as a static network. Such representation focuses on the strategic infrastructure developed that form the network. Such infrastructure are road/ highways, rail and river ways, intermodal terminals and even freight handling airports. Such infrastructure has fixed location, and design properties (e.g. number of rail tracks, road lanes, warehouse area) that can be associated to static properties.

For operational level network representation more dynamic properties require to be considered. As discussed in the previous section, queues are a typical issue in logistical chains. Regardless of weather a queue will be represented as an integer number of queueing items or as an expected travel time on a link, this is a dynamic property that frequently requires updating. Road mapping services, typically offer live congestion information, and by using the link travel time representation, can estimate the accumulated impact of queues, on any route.



Figure 5.6: PI data structure classification

Therefore, the PI network data structure considers data for nodes, links and movers, involving both their static properties and their dynamic status. As illustrated in Figure 5.6, the data model aligns with the GPICS definitions, but also functions as an open ended consolidated information service, where every user can potentially identify relevant and useful information for routing and synchro-modality decision making.

5.1.2.1 PI Nodes

For PI Nodes static data involve the location, the intermodal connectivity, the warehousing capability/ capacity and the functions (e.g. intermodal, cross-docking). Dynamic information include the spare warehouse capacity and the queue information per service provided. Additional information on weather might also be relevant in case a PI Order involves weather conditions in the conditions of carriage.

5.1.2.2 PI Links

For PI Links the static data include the origin and destination nodes locations and IDs, the mode, the functions (e.g. refrigeration capability), the ride quality (bumpiness) and the distance . It may be the case that there are two or more links between the same origin and destination if more than one modes are available. Several dynamic weights can be associated to PI Links including:

- Cost, that may simply consider the link length and the per km cost of the link's mode. Additional detail can be added to account for staffing costs (e.g. driver) in associationg to travel time.
- Travel time, represents congestion levels as well as queuing that occurs at the link.
- Weather conditions, represents and average expected conditions during the time of use, that reflect parameters in the conditions of carriage
- Frequency, represents the frequency of services between two nodes. This may be an indicator of the expected waiting time of a container upon arrival at a node. This weight can be useful in calculating adverse routing scenarios.
- Reliability, represents the chance that the delivery will be made in a safe and timely manner. This metric, although it may contain bias on other metrics, is a useful in enabling shippers to set limits for the conditions of carriage.
- Congestion, represents the travel time loss, and can be useful to identify synchro-modality opportunities
- Emissions, represents the CO2 emitted depending on the distance and mode of a link. Lastly,
- Consolidation represents the expected level of consolidation on that link. This weight is based on historical data, and influences the emissions, and cost weights depending on the routes average fill rate. Therefore, a less polluting mode, with a low consolidation level, might be a less environmentally friendly option, than a more polluting mode with higher consolidation level.

5.1.2.3 PI Movers

PI Movers can be in the form of line-haul services or ad-hoc jobs. Line haul services offer more opportunities for consolidation, rather than assigning the carriage of goods between two nodes directly to a PI Mover. Information on the location (and more generally availability) of non line-haul trucks and trailers, their services and capacity are essential for handling cases where line-haul services cannot be utilised. For line-haul services, the link, timetabling, and capacity per service static information are considered. In term of dynamic information, the PI Movers fill rate is required, to enable the PI capability for synchro modal routing of cargo, and achieve improved operational efficiency. Reliability and frequency of service weights can also be considered, in the same context as in the case of PI Links.

An additional element of the PI digitisation, that is backed by the proposed data structure, is that PI Orders information on the condition of carriage (such as weather condition limitations, mode limitations or penalties for late delivery or damage) need to be associable with network data.

5.1.3 The Transportation Planning Manager

As planning for transport involves future actions, a preliminary analysis of the network and the available transport options must be carried out first. The type of available transportation services both according to the type of transport (e.g. refrigerated truckloads, open rail cars, etc.), the geographical areas (shipping lanes/transport corridors) serviced by the carriers, and the prices charged for each

transport mode and service within each network path must be considered and analysed. Additional constraints posed by the shipping instruction such as time windows, must be taken under consideration also of network characteristics such as capacity limitations of hubs and distribution centres, preferred carriers, and so on. In order to achieve an optimal planning solution, a transportation planning manager therefore must balance the different requirements and constraints with the availability options.

The transportation planner deals with the short term/ operational functionality of the networking layer and acts upon a shipping instruction received by the shipper. The transportation planner will analyse the shipping instruction and decide on the best way to forward it through PI by considering the available and feasible network options. The transportation planner needs to work with the shipper, the carrier who will transport the goods (at least to the initial PI hub) and also possibly with the terminals/hubs themselves.

Pricing, timing, capacities, and quality of service drive the decision of what modes and routes the transport planner will consider use. PI Links and Hub rates (costs), capacity, and associated efficiencies (time and service levels) will determine the competitiveness of each corridor. Issues like congestion and backhauling must also be taken into account.

5.2 Networking Service Protocol

To accommodate the dual functionality of the Networking Service, it is divided into two separate stages.

- Stage 1 deals with the collection and integration of PI network information, and
- Stage 2 deals with the provision of PI Shipment specific information.

5.2.1 Network discovery module

In order to identify the relevant nodes and links of the network(s) in collaboration with the Link and Physical Layers, the Networking service Stage 1 classifies them in terms of geographical location (and scale), transport mode, and level of aggregation as proposed by the GPICS PI Node and PI Link typology.

The standardised classification of the PI components enables the interoperability across various services and systems that communicate with the Networking Service. Data are refreshed depending on their nature (static, dynamic, live) at different frequencies. For example static data, that capture infrastructure changes (new roads, new ports. Hubs) are updated every few days. Dynamic data such as weather conditions, road works, or truck fill rates are updated every few hours. Finally, live data, that concern travel times, queues and congestion are updated every few minutes.

The Networking Service seeks the required data from various data sources as illustrated in **Figure 5.7**. At the same time, it collects and analyses the data from ICONET IoT platform to cross check information. Finally, it holds a database of historical weight values, that it uses to make short term predictions for weights such as PI Link consolidation rate. This capability can be useful in helping the transport planner tool or the routing service to make enhanced and more complex routing decisions.



Figure 5.7: Networking service network discovery (Stage 1)

5.2.2 Data packaging module

Networking Service Stage 2 aim to pack information relevant to a specific PI Order. This module is of little value when the datasets are small, but it becomes increasingly useful as the PI grows in scale and detail. The data structure presented in this Chapter is utilized throughout the application of the Area, Modes, Aggregation level and Data detail tools, which are described as follows:

The aim of the geographical scale function (Area tool) is to limit the scope of the search area for network components. An area of relevance is identified on the global map, for the specific PI shipment submitted. For example, the scale will be different for a request to carry a cargo from North to South Europe, or between two neighbouring French cities. The Area tool utilizes the origin and destination coordinates of the PI shipment to identify an oval shaped area of relevant PI network components.



Figure 5.8: Interactions of Networking Service data packaging tool

The mode tool considers restrictions imposed by each PI order on the modes available for shipment. If more than one modes are feasible, the Networking Service assess the multimodality options available, by considering transhipment nodes and various mode links.

The aggregation tool considers the level of detail required for the routing request associated to a specific PI order. PI nodes can represent international or local hubs in the case of long-haul shipments, local warehouses and postcodes in the case of e-commerce, as well as specific function (e.g. customs) in complex port (intra-hub) operations.

With awareness on the scale, aggregation level and modes a final decision is made on the level of data detail required. This will depend on the PI order made, but also on data availability. The output data detail can range from physical properties of infrastructure, to live information on the services operating on the PI network. Four levels of data details can be identified:

- Infrastructure properties: Network information describe static infrastructure characteristics such as the length of a link, the modes that can accommodate the function of carrying cargo (e.g. truck, rail), or even more detailed information such as classification into motorway, or number of lanes. A similar concept can be applied to the description of nodes. A node can represent a warehouse that has specific capacity for storage and docking capability.
- Operational status: Additional to network static information, operational condition information is collected for each PI network component. For example, incapacitated links due to roadworks or traffic, will have their travel time and travel cost (if applicable) updated with most current information. Similarly, for nodes up to date availability and status of infrastructure is considered, such as spare capacity.
- Services schedule: The aim of this data layer, is to account for the fact that roads and warehouses do not handle directly cargo transport, but rather indirectly through services. Truck, air, river/sea or rail transport hauling services that utilize the infrastructure are responsible for undertaking the task of physically moving cargo from one location to another. With that in mind, at Stage 2/ Step 1 the Networking Service collects (if available) service schedules that operate between specific locations.
- Services status: A final layer of complexity can be anticipated, that accounts for live information on the capacity of en-route hauling services and queues at PI hub services.

It is also essential to associate each physical infrastructure component with properties to be used for operational decisions. Networking Service Stage 2/ Step 2, deals with this requirement through four tools that adjust the information collected in Step 1 to specific PI Order requirements.

- Link filter and node filter tools capture the restrictions imposed by the PI Order and filter out any network components that do not meet them. For example, for a shipment that requires refrigeration, the respective PI node and PI service property is examined, and nodes and services that do not offer this option are not considered further.
- The Link and Node Cost tools calculate the final metrics for each PI component in terms of KPIs as requested in the PI order, such as travel time, travel cost, service reliability. Performance is adjusted for each PI component considering the requirements, constraints and limitations implemented in the previous steps of the Networking Service.

The final output of the networking service is a set of interconnected PI nodes, PI links and PI services, that are available for transporting a shipment between any two nodes. Stage 1 of the Networking Service that is responsible for collecting information on the status of the network, is always listening for changes in traffic or services status. Depending on the nature of the PI Order the Networking Service Stage 2 can be either called once or several times. For cases that only static information are available the Networking Service is called only when an order is initiated, while for cases that information is dynamically updated, it is called whenever a shipment arrives at a PI node that it is not the destination. **Figure 5.8**, captures the operational sequence for Stage 2 of the Networking Service.

5.2.3 Optimal fulfilment store assignment to orders

In the context of urban delivery, it is often the case that the destination and time slot of an order is known, but there is no given order fulfilment origin. To incorporate urban eCommerce delivery into the concept of the Physical Internet, an fulfilment centre identification tool has been incorporated into the Networking Service.

The aim of the tool is to associate orders with unknown fulfilment locations to optimal origin locations. This is handled through a linear optimisation model that seeks to minimise the todal distance for satisfying all orders. Assuming that the known distance d_{ij} between every customer location j and every fulfilment store i, and that an additional integer variable s_{ik} captures the stock of products available at each fulfillment store i per product k. And an additional integer variable o_{jk} captures the number of products of product k ordered in order j. A binary decision variable x_{ij} is equal to 1 if fulfillment store i is chosen to satisfy customer order j, and is 0 otherwise. Then, a cost minimization problem can be formulated as follows:

Objective function:

$$\min_{x_{ij}} \sum x_{ij} d_{ij}$$

Subject to constraints:

$$\sum_{i} x_{ij} \ge 1 \quad \forall j, k$$
$$\sum_{j} x_{ij} * o_{jk} \le s_{ik} \quad \forall i, k$$
$$x_{ij} \in \{0, 1\}$$

The first constraint ensures that all orders are satisfied. The second constraint ensures that the store capacity for each SKU is not exceeded, while the last constraint defines the possible values for the decision variables.

5.3 Service sample application and design guidelines

Networking is the group of processes and activities that analyse the available network options for transporting the goods to the destination, according to the shipping instruction.

5.3.1 TEN-T PI Network

In the context of the North-South European corridor Living Lab, the Networking Service function is to undertake the network discovery and share the information with other services. The network illustrated in **Figure 5.9**, represents the nodes of the TEN-T network (green points) and the P&G warehouse locations (red points). PI Links of road, rail, and river modes are considered between PI Nodes, and a road link is introduced between every P&G warehouse and its closest PI Node. Sea links are considered for the PI Nodes that are not accessible otherwise (e.g. Cyprus).



Figure 5.9: North-South Europe PI Corridor

Although P&G does not consider penalties for late delivery of their Tier 1 (high urgency) shipments, it has asked freight forwarders to provide separate rates for 5% express delivery, or rates including 5% express. The PI should therefore be able to provide customized delivery features with respect to the speed of delivery. For example urgent, low cost, lost emissions.

linkid	linkorigin	linkdestination	originid	destinationid	linkmode	costeuro	distkm	traveltimemins
0	Lefkosia	Limassol	148	147	Road	127	84.0	60
1	Limassol	Athens	147	145	Sea	48	972.3	2100
2	Athens	Patras	145	146	Road	319	210.0	141
3	Athens	Patras	145	146	Rail	48	210.0	157
4	Athens	Thessaloniki	145	143	Road	761	501.0	305

Figure 5.10: TENT PI Links and their weights

By adopting the multiple weight data structure illustrated in **Figure 5.10**, information on the distance, cost, and travel time can be retrieved by routing services for calculating the optimal route to the destination. The optimal route identified changes as different weights are considered. Furthermore, the weights can be customised per customer needs, as in the case of P&G to deliver cargo with different priority.

5.3.2 eCommerce

In the context of eCommerce and urban deliveries, the information required for the shipment of cargo differ when compared to the generic PI case of long haul shipments. To account for this unique feature of eCommerce, the description of PI Nodes has been extended to accommodate product stock level's as illustrated in **Figure 5.11**. Operating hours and picking capacity are considered as service properties, that are also defined in generic PI Nodes. This representation enables the identification of shipment origins if this is not known, as discussed in Section 5.2.3, where customer orders were associated optimally to order fulfilment stores.



Figure 5.11: eCommerce PI Node information structure

5.3.3 Design guidelines

To develop best-practice guidelines the end-to-end process of the Networking Service in the context of each Living Lab is examined. With each LL offering a substantially unique point of view into the Physical Internet, Networking Service are proposed for addressing the design requirements in each Living Lab.

5.3.3.1 PI Hub

In the context of a PI Hub the PI Networking Service considers a network that spans from the arrival and departure terminals offered by each mode (sea international and intercontinental trade, road, rail river connections to hinterland). In terms of network representation aggregation, the PI networking service captures port and regional intermodal hubs, but also port services that cargo needs to transverse, such as customs, or port stack capacity. Such artificial nodes are associated with throughput rates or throughput capacity, that are frequently the cause of port congestion and queues. Additional routing options within the port are explicitly represented by Links. The data include both the static properties and dynamic operational status information of the network, considering intermodal capability where available. Localised failures or congestion are anticipated to inform and trigger cargo re-routing options. Such a networking service enables, the PI to provide enhanced modal choices that align with the hub's infrastructural capability. Furthermore, it has a long-lasting effect on the efficient utilization of port capacity and facilitates.

5.3.3.2 PI Corridor

The Networking Service design for PI Corridors spans geographically across the European continent providing several north-south corridors. The TEN-T network nodes (Level 1 GPICS) are considered as inflow or outflow points, as well as points of consolidation or transhipment provided sufficient infrastructure is in place. The modes serving adjacent links capture not only for road, rail and river that

stretch inland, but also for sea links such as Norway-Finland and Italy-Greece. Considering the scale of the network, aggregation focuses on the city nodes and links of the network. Each node is associated with a certain level of production and consumption. For intermodal nodes, throughput capacity of intramodality is also considered. To enable synchro-modality and cargo consolidation at nodes, data covering infrastructure properties, status, as well as services schedule and loading status are maintained. This enables the PI Corridor, to achieve higher utilisation rates of infrastructure and services.

5.3.3.3 eCommerce

In eCommerce the distribution network is typically operated by Logistic Service Providers. In such cases, intermodal transhipment options are not considered. Network aggregation considers the capacity of each regional store to accommodate contingency stock and the capacity of each vehicle to consolidate deliveries, in the sense that it can pick up a contingency stock along the way to satisfy demand at an outlet approaching stock-out. The networking service optimal fulfilment store identification module is also deployed, if only less sophisticated approaches are in place. To achieve consolidation of shipments, a temporal representation of vehicle loading capacity availability is maintained based on dynamic service loading data.

5.3.3.4 WaaS

Scale-wise the Warehouse as a Service (WaaS) focuses on the region the specific providers facilities cover. This is typically national, hence a national scale model, that also considers national points of cargo entry and exit is considered. As both rail and road links are considered, national entry and exit points should also account for France international rail freight hubs. With facilities and customers spread all across the coverage region, a detailed network of links, supplementing the TEN-T network is maintained, capturing the distances between all warehouses, as well as customer pick-up and delivery locations. WaaS nodes are represented at highly disaggregate level, to incorporate warehouse processes for incoming and outgoing storage, as well as cross-docking services. Data-wise the properties, services and storage availability status of each warehouse is maintained.

The design recommendations made for each Living Lab context are summarized in Figure 5.12.



Figure 5.12: PI Networking design recommendations

6 ROUTING SERVICE

This section reports on the approach adopted for PI routing service of ICONET. The routing service builds upon the routing algorithm proposed in first and second versions of this deliverable (D2.3 & D2.4). The objective of routing service is to find the best path between two PI-Hubs considering travel time, distance and CO2 emission costs for a given set of constraints. The previous version (D2.4) of this deliverable, formulated the routing service as Vehicle Routing Problem (VRP) and Travelling Salesman Problem (TSP) with some modifications to consider the aspects of the PI such as intermodal transportation and real-time changes in the PI network. In this deliverable, we additionally report on advancements made in implementing the solutions for VRP/TSP. Note TSP is subset of VRP and deals with one route whereas VRP can handle multiple routes. Travelling Salesman problems (TSP) and its more generic form Vehicle Routing Problems (VRP) are classic combinatorial (NP-hard) problems in operations research (OR) and these are formulated as integer constrained optimization, i.e. with integral or binary decision variables. The theory and algorithm design communities have typically used graphs to formulate these problems. Classically, approaches to tackling an optimization problem can be categorized into exact, approximation, and heuristic algorithms. Exact algorithms are based on enumeration or branch-and-bound with an integer programming formulation and they guarantee to find optimal solutions but are not feasible for large instances. On the other hand, polynomial time approximation algorithms are tractable for large instances, but may suffer from weak optimality guarantees or empirical performance. Finally, heuristics are fast and effective algorithms but require problem-specific knowledge and manual design of mathematical model for the solution.

To reduce the effort of manual mathematical modelling for solutions to OR problems optimization, researchers have recently, especially after the seminal work of Pointer Networks (Vinyals et al., 2015), started looking at machine learning and reinforcement learning based approaches (Kool et al., 2019; Nazari et al., 2018; Bengio et al., 2018; Khalil et al., 2017; Bello et al., 2016). The learning based methods are trained on large number of problem instances, and have been shown to be extremely fast in producing solutions of reasonably good quality. However, when compared with the same benchmark instances, these learning-based methods cannot outperform the state-of- the-art method LKH3 (Helsgaun, 2017), which is a penalty-function-based extension of classical Lin-Kernighan heuristic (Lin & Kernighan, 1973).

In the ICONET project, we attempt to combine the ability to search for accurate solution by heuristic algorithms with the ability of machine learning to learn from large number of instances. In particular, we combine the strengths of both worlds in a framework while keeping the focus on intermodal and dynamic aspects of the PI. Therefore, we propose an end-to-end framework that learns solution dynamically in an iterative fashion instead of directly constructing a solution to a TSP/VRP instance. This is based on the approach known as *neural combinatorial optimization* proposed by Bengio et al., 2018.

- We present a novel modular framework where instead of all operators of a solution in one algorithm, it combines pluggable modules to learn an approximate solution directly from the problem instance.
- The framework can learn approximate solution form a huge number of problem instances in a much faster way than the heuristic methods while optimal solution is derived via supervised or reinforcement learning.

• Learning is based on Graph Neural Networks (GNN) where we represent nodes and links in embedding space through convolutions layers in a temporal fashion to capture the dynamic changes in the PI network.

6.1 Routing Service Considerations and Design

In the previous version of this repot (D2.4) a detailed description of the approach to modelling the PI Network movements and deploying the Vehicle Routing Problem as well as its variants were discussed into detail. This section focuses on computational complexity and solution algorithms for routing problems.

6.1.1 Computational complexity of optimal routing algorithms

Due to the large scale and number of components participating in the Physical Internet, in the following sections, we investigate the trackability and complexity of routing computation algorithms. Heuristic and learning based routing solutions are investigated that have been implemented and/or adapted for the LL scenarios in the ICONET project

6.1.1.1 Heuristic Solutions for PI Routing

Simplest algorithm which guarantees the solution TSP problem is by generating all possible tours of the nodes in the graph and choose the shortest tour. However, such an algorithms is not feasible for large number of nodes. The reason being the large number of node permutations that need to be exhausted before finding a combination of nodes with minimum path length. This can be explained by a simple calculation. Let's assume, it takes 2 seconds to determine a TSP tour as solution for visiting 10 nodes; i.e. exact algorithm will have to go through 10! possibilities to find best solution in terms of minimum path length. Now if one more node are added then on same computing machine it will require 2 secs × 11! / 10! \approx 22 secs. Adding one more node will require 2 secs × 12! /10! \approx 4 mins. And for n=14 time complexity increases to 2 secs × 14! / 10! \approx 13 hours, for n = 18 it will take 2 secs × 18! / 10! \approx 112 years and so on. It is evident from these simple calculations why finding exact solutions are not feasible for TSP. This requires approximation of the exact solution. In other words, we find the path with minimum distance but algorithm does not guarantee if the solution is the only shortest. The following sections give details for some of the heuristic approximation algorithms, we used to solve TSP route in the ICONET project.

6.1.1.2 Alternating Nearest Neighbour with Repetition

Nearest neighbour (NN) algorithm starts from an arbitrary initial node and repeatedly chooses next unvisited node. The final tour gets extended at each step by including a nearest unvisited node and finally return to start node.



Figure 6.1: Nearest neighbour algorithm

In order to determine best NN algorithm for a given problem, we can compare performance of NN by using each of the nodes in the graph as start node and determine the optimal start point. To compare NN performance we determined the optimality gap for NN from exact algorithm. This is achieved by determining the ratio of tour length found by NN with tour length from the exact algorithm:

 $Optimality_Gap = \frac{TourLength_{NN}}{Tourlength_{EXACT}}$

Generally, design of heuristic algorithms may consist of one or more of the following options:

- 1. Tour construction algorithms
- 2. Tour improvement algorithms
- 3. Ensemble algorithms

So far we have constructed the tour with NN algorithm. In order to further optimize the solution, we adopt a strategy to improve the solution. One strategy is the *repetition strategy* under which algorithm is re-run multiple times varying some aspects (such as start node in our NN algorithm case) and solution with best score is selected as optimal solution. Second strategy is *alteration strategy* where initial constructed tour is further improved by making changes.

One such improvement algorithm is 2-opt algorithm: Start with a given tour. Replace 2 links of the tour with 2 other links in such a way that the new tour length is shorter. Continue in this way until no more improvements are possible. One of the most effective approximate algorithms for TSP is *Lin-Kernighan (LKH)* which adopts *k-opt* strategy for improvement. Finally, tour construction and tour improvement algorithms can be combined in an ensemble strategy which we adopt in designing our algorithms for routing service in the ICONET project.

One of the shortcomings of NN algorithm is if there are outliers in the data. For example, in a scenario where one location/city to be visited is far away from rest of the cities. In such case NN algorithm will visit cities which are close to each other before visiting the city which is outlier. In this case, there is overhead in visiting outlier locations. To address this, we determine final tour by reversing the segments where a segment is a sequence of consecutive cities within a tour. A segment is open-ended and does not have loop. So if [A, B, C, D] is a tour then one of the segments combinations defining this tour can include [A, B, C], [C, D]. In the implementation, we reverse only if it decreases the path length.

Algorithm: improve t	our by segmentation
<pre>Function improve_tour Input: tour, N Original_length = ler Function find_all_pos for start, end in all reverse_tour if length(rev retur return tour</pre>	gth(tour) sible_segments(N) . segments = reverse segment(tour, start, end) verse_tour) < length(tour) rn improve_tour

Figure 6.2: Improve tour segmentation algorithm

6.1.1.3 Greedy Algorithms

Greedy algorithms at every step add to the solution tour the edge that is shortest. The algorithm maintains a set of segments. Initially each node in the graph is a segment on its own. Then nodes from different segments connect with each other with shortest possible edge. Algorithm repeats until a tour including all nodes is formed.



Figure 6.3: Greedy algorithm

6.1.1.3.1 Results & Discussions

For experiments, a set of nodes with a tuple (x, y) as coordinates and distance between points is the Euclidean distance i.e. abs(node1 - node2). We used random generator with seed to assign a random value between 0 and 1000 as distance between nodes. The seed was used to produce same values for different experiments. Our experiments included the nearest neighbour heuristic with repeated and altered strategy. Experiments were performed on a 2.6GHz Intel Core i9 Processor with 16GB RAM.

Method	N=10		N=11		N=12		N=14	
	Avg. Tour Time		Tour	Time	Tour	Time	Tour	Time
	Length		length		length		length	
Exact	2229.1	1.243s	2196.2	9.72s	2482.4	138.79s	NA	NA
Solution								
NN	2381.4	0.00s	2404.1	0.00s	3241.8	0.00s	3283.2	0.00s

Table 6.1: Greedy vs Nearest Neighbour (NN) heuristic

And for approximate NN and its variants

Method		N = 10		N = 20		N = 50		N = 100	
		Avg. Tour	Time	Tour	Time	Tour	Time	Tour	Time
		Length		length		length		length	
NN		2381.4	0.00s	3363.5	0.00s	5039.5	0.00s	6734.1	0.001s
NN	Repetition	2297.7	0.00s	3218.8	0.001	4511.7	0.018s	5912.6	0.118s
Strategy									
NN 2-opt strategy		2333.4	0.00s	3076.8	0.001s	4408.8	0.003s	5909.5	0.018s
NN	Ensemble	2291.8	0.002s	3022.1	0.015s	4169.1	0.094s	5701.6	0.364s
strategy									
(Repetition + 2-opt)									

Table 6.2 Comparison of optimization strategies for NN

First of all, we notice in Table 6.1 that exact algorithm is not able to compute a solution for a graph with n>=14 in a reasonable time. In Table 6.2, we see that for we are able to find solutions for larger TSP instances. In order to see the difference between both algorithms, we point reader to above tables when n=10. For n=10 exact algorithms find solution with shorter length but in plain NN case algorithms is faster, but it did not find shortest tour. The optimality gap for n=10 over 10 training samples is given below:



Figure 6.4: Optimality gap

The ratio of 1.0 means NN and exact algorithms got the same (optimal) result; that happened just once times out of 10. The other times, we see that the NN produces a longer tour, by anything up to 34% worse, with a median of 1% worse.

When NN compared with its variants with different repetition and alteration strategies, we see in Table 6.2 that NN with repetition strategy performs better than the vanilla NN and close to exact algorithm. In order to determine the number of repetitions required to find best solution, we run the algorithm for 30 instances with 60 to 150 starting nodes where optimal tour is determined over a range of 0 to 100 repetitions, We record average, min, max of tour length and their standard deviations from the mean.

NN variants	Mean	Min	Max	Std. deviation	Time per tour
NN (reps=0)	7195	6315	8180	441	0.001s
NN (reps=10)	6753	6325	7529	336	0.011
NN (reps=20)	6673	6238	7462	289	0.019
NN (reps=50)	6595	6223	7243	255	0.047
NN (reps=100)	6575	6213	7243	259	0.086

Table 6.3: Benchmarking for starting nodes and repetitions

From Table 6.3, it is clear that NN algorithm results in shorter average tour length but we start getting diminishing returns after 50 repetitions. It will depend on the use case priorities (run time versus tour length), somewhere around 25 or 50 repetitions might be a good trade-off.

In addition to NN and NN with repetition, Table 6.2 shows the results from NN with alteration and ensemble strategies and we see that average tour length shortens with ensemble strategy but run time increase. So depending on the LL's scenario, we will adopt strategy accordingly in the ICONET project.

6.2 Routing Service Protocol

6.2.1 Service Design

6.2.1.1 Deep Learning based PI Routing

Deep Learning models have led qualitative breakthroughs with Euclidean data such image, text and speech for a wide variety of tasks such as speech recognition, machine translation and image analysis. Convolutional neural networks (LeCun et al., 1998) are generic building blocks for deep learning architectures for Computer Vision and NLP tasks, but ConvNets require regular data such as 2D and 3D grids for computer vision and 1D text sequence for NLP. However, in combinatorial optimization problems, real-world data has irregular structure and is non-Euclidean. Supply chain networks, transport networks, and sensor networks are examples of non-Euclidean data structure and can be modelled as graphs.





6.2.1.2 Problem formulation

One of the special cases of VRP is when there is only one vehicle under consideration. Such scenario can be imagined when routing PI-Containers that have same destination and route optimization is required by PI-mover transporting those containers. This simple case of VRP is Travelling Salesman Problem and in this deliverable, we formulate VRP or TSP as a learning problem on graphs. Formally, given a fully connected PI graph of n PI-Hubs in two dimensional unit square $S = \{x_i\}_{i=1}^n$ where each $x_i \in [0,1]^2$, the objective is to learn a permutation of the nodes p as a tour that visits each node once and has minimum total length, defined as:

$$L(p \mid s) = ||x_{p_n} - x_{p_1}||_2 + \sum_{i=1}^{n-1} ||x_{p_i} - x_{p_{i+1}}||_2$$

where $\|.\|_2$ denotes l_2 norm.

6.2.1.3 Logistics Network Encoding

A graph neural network (GNN) encoder computes *d*-dimensional representation of nodes (PI-Hubs in ICONET) in the graph (PI-Network in ICONET). In its most basic form, an encoder is a function that maps a node directly into a vector space. Formally, the encoder is a function, $ENC : V \to \mathbb{R}^d$, that maps nodes to vector embeddings, $z_i \in \mathbb{R}^d$ (where z_i corresponds to the embedding for node $v_i \in V$). The state-of-the-art methods that imply such encoding techniques are based on deep neural network architectures and have shown best performance in link predictions tasks (Hamilton et al., 2017a; Kipf and Welling, 2017; Pham et al., 2017). At each layer of the GNN, nodes aggregate features from their neighbouring nodes to represent local graph structure via recursive message passing (Gilmer et al., 2017). With the message passing scheme node and edge features at layer $\ell + 1$ can be represented in terms of node features (h^ℓ) and edge features (e^ℓ) at layer through linear transformation as:

$$h_i^{\ell+1} = h_i^{\ell} + f\left(BN\left(U^{\ell}h_i^{\ell} + AGGR_{j \in \mathcal{N}_i}\left(\sigma\left(e_{ij}^{\ell}\right) \odot V^{\ell}h_j^{\ell}\right)\right)\right)$$
$$e_{ij}^{\ell+1} = e_{ij}^{\ell} + f\left(BN\left(A^{\ell}e_{ij}^{\ell} + B^{\ell}h_i^{\ell} + C^{\ell}h_j^{\ell}\right)\right)$$

BN is BatchNorm (loffe&Szegedy, 2015) normalization, AGGR is neighborhood aggregation function such as SUM, MEAN or MAX. f is activation function such as ReLU, σ is the sigmoid function, and $A^{\ell}, B^{\ell}, C^{\ell}, W^{\ell} \in \mathbb{R}^{d \times d}$ are learnable parameters. In ICONET, we adapt encoders based on two different deep learning architectures.

6.2.1.3.1 Encoder - Graph Convolution Network

Here, we give the details of Graph Convolution network (GCN) architecture adapted as encoder for ICOENT's PI graph. As input layer to GCN, we convert two dimensional coordinates of PI-hubs to h-dimensional node and edge inputs for n nodes and n^2 edges respectively. The node inputs h_i are computed via simply linear transformation. To compute the edge inputs e_{ij} for an edge between nodes i and j, we first compute the distance matrix D, where d_{ij} corresponds to the Euclidean distance between PI-Hubs i and j.

Graph Convolution layer is a residual Gated Graph Conv net as proposed by Bresson & Laurent (2017). The key aspect of this architecture is the edge gating mechanism which enables the model to learn importance for edges in the TSP or VRP problem.

$$h_i^{\ell+1} = \operatorname{ReLU}\left(U^{\ell}h_i^{\ell} + \frac{\sum_{j \to i} \eta_{ij}^{\ell} \odot V^{\ell}h_j^{\ell}}{\sum_{j \to i} \eta_{ij}^{\ell} + \epsilon}\right)$$

 η_{ij}^{ℓ} are the edge gates, $\eta_{ij} = \sigma (A^{\ell} h_i^{\ell} + B^{\ell} h_j^{\ell})$. A^{ℓ} and B^{ℓ} parameters learned at edge gates are then used to compute edge feature vector e_{ij} .

 ϵ is a small positive constant to avoid division by zeros, and *ReLU* is the rectified linear unit (*ReLU*(*z*) = max(0, z)) applied element-wise to its input.

The final layer in the model architecture is Multi-Layer perceptron classifier used to compute the probability of edge embeddings e'_{ij} for being connected in the final TSP/VRP tour of graph. For example, *softmax* operation can be used with 2-layer perceptron as follows:

$$w_{ij} = \text{softmax} \left(\theta_1 \text{ReLU} \left(\theta_2 \ e'_{ij} \right) \right)$$

6.2.1.3.2 Encoder - Graph Attention Network

We also experimented with another encoder with Graph attention Network layer following Transformers (Vaswani et al., 2017) architecture. In this architecture, attention mechanism is used as weighted message passing between nodes in the graph. One of the benefits of attention mechanisms is that they allow for dealing with variable sized inputs, focusing on the most relevant parts of the input to make decisions. The weight of the message value that a node receives from a neighbor depends on the compatibility of its query with the key of the neighbour as shown in **Figure 6.6**.



Figure 6.6: Attention mechanism. Image source (Kool et al., 2019)

$$\mathbf{q}_i = W^Q \mathbf{h}_i, \mathbf{k}_i = W^K \mathbf{h}_i, \mathbf{v}_i = W^V \mathbf{h}_i$$

Compatibility of node *i* with *j* is computed as:

$$u_{ij} = \begin{cases} \frac{\mathbf{q}_i^T \mathbf{k}_j}{\sqrt{d_k}} & \text{if } i \text{ adjacent to } j \\ -\infty & \text{otherwise} \end{cases}$$

From nodes compatibilities, attention weights are computed

$$a_{ij} = \frac{e^{\omega_i j}}{\sum_{j'} e^{u_{ij'}}}$$

Finally, the \mathbf{h}'_i vector is received by node *i* is combinations of messages v_i .

$$\mathbf{h}_i' = \sum_j a_{ij} \, \mathbf{v}_j$$

In the implementation of attention mechanism, we adapted the architecture proposed in (Kool et al., 2019), where each attention layer consists of two sublayers: a multi-head attention (MHA) layer that executes message passing between the nodes and a node wise fully connected feed-forward layer. Each sublayer adds a dropout and a batch normalization (*BatchNormalization* (BN) used here). The MHA layer uses 8 heads and the feed-forward sublayer has one hidden sub-layer with dimension 512 and *ReLu* activation.

$$\begin{aligned} & \hat{\mathbf{h}}_{i} = \mathrm{BN}^{\ell} \left(\mathbf{h}_{i}^{(\ell-1)} + \mathrm{MHA}_{i}^{\ell} \left(\mathbf{h}_{1}^{(\ell-1)}, \dots, \mathbf{h}_{n}^{(\ell-1)} \right) \right) \\ & \mathbf{h}_{i}^{(\ell)} = \mathrm{BN}^{\ell} \left(\hat{\mathbf{h}}_{i} + \mathrm{FF}^{\ell} (\hat{\mathbf{h}}_{i}) \right) \end{aligned}$$

6.2.1.4 Solution Decoding

Decoder outputs the next node in the solution based on embeddings of node from the encoder and the outputs generated at time t sequentially. Each sublayer has two sub-layers of MHA and fully connected feed-forward. Similar to the encoder, each sub-layer adopts residual connection and a layer *BatchNormalization*. The key element in the decoder is the context embedding vector. The context at time t consists of the embeddings of the graph, the previous node at t-1 and the first node. Essentially, we follow the attention decoder from proposed in (Kool et al., 2019), which starts from a random node and outputs probability distribution over its neighbors at each step. At time step t at node i, the decoder builds a context \hat{h}_i^C for the partial tour $\pi'_{t'}$, generated at time t' < t, by packing together the graph embedding hG and the embeddings of the first and last node.

$$\hat{h}_{i}^{C} = W_{C}\left[h_{G}, h_{\pi_{t-1}^{\prime}}^{L}, h_{\pi_{1}^{\prime}}^{L}\right]$$

The context is then refined via a MHA over node embeddings:

$$h_i^C = \text{MHA}\left(Q = \hat{h}_i^C, K = \{h_1^L, \dots, h_n^L\}, V = \{h_1^L, \dots, h_n^L\}\right)$$

6.2.1.5 Solution Search

In order to search for solution for TSP or VRP, one approach is to start from first node and then we select a node from its neighbours based on the highest probability from decoder; i.e. *greedy search*. If our decoder is based on MLP on node embeddings produced by the final GNN encoder layer L, we can compute unnormalized edge logits:

$$\hat{p}_{ij} = W_2 \left(\text{ReLU} \left(W_1 \left(\left[h_G, h_i^L, h_j^L \right] \right) \right) \right)$$
, where $h_G = \frac{1}{n} \sum_{i=0} h_i^{L_i}$

The logits \hat{p}_{ij} are converted to probabilities over each edge p_{ij} via a softmax function. Since the probabilities are independent of each other, here we can obtain a valid TSP/VRP tour using greedy search to traverse the graph starting form a random node and masking previously visited nodes.

To further improve the quality of solution beam search can be used. A beam search is a limited-width breadth-first search. For a sequence to sequence model, a beam search expands at every step t = 0,1,2,... at with width b partial sequences with highest probability to compute the probabilities with length t + 1. Similarly, we can sample b solutions from the decoded solutions and select the shortest tour among them.

6.2.1.6 Policy learning

The final step adapted approach is policy learning. There are two ways to learn optimal solution policy; one is through supervise learning where decoder output is compared with ground-truth and crossentropy loss is minimized for each edge prediction in the final solution. Second method is Reinforcement learning (RL) in the absence of ground truth. With RL, models can be trained by minimizing the tour length via policy gradient algorithms (Sehnke et al., 2010) such as REINFORCE (William, R.J. 1992), Actor critic, and A3C or Q-Learning (Khalil et al., 2017). We adapt REINFORCE (also know as Monte-Carlo policy gradient) for policy learning due to its simplicity to learn policy directly with a parameterized function θ as $\mathcal{L}(\theta \mid s) = \mathbb{E}_{p_{\theta}(\pi \mid s)}[L(\pi)]$, the expectation of the tour length $L(\pi)$, where $p_{\theta}(\pi \mid s)$ is the probability distribution from which we sample to obtain the tour $\pi \mid s$.

6.2.1.7 Experiments

Our experiments are limited to TSP and two variants of VRP; CVRP and SDVRP. For TSP we implement both Convolution and attention-based models (Graph ConvNet and Graph Attention Net resp.) and for the VRP our implementation is limited to attention model only. First, data is simulated for TSP and VRP and objective functions are defined (see D2.4 for details on objective functions for different variant of VRP). The models are trained for variable graph sizes with 10, 20, 30 and 100 nodes which are the conventional size to benchmark TSP or VRP in the literature. For the both problems, we use same hyperparameters.

6.2.1.7.1 Hyperparameters Configuration

We implemented two types of configurations:

- 1. Three GNN encoder layers followed by attention decoder head with 128 hidden dimensions.
- 2. Four GNN encoder layers followed by *softmax* edge predictor.

We use a constant learning rate $\eta = 10^{-4}$. The best configuration was Graph ConvNet encoder with MAX aggregation and BatchNorm followed by attention mechanism decoder. All the models are trained via supervise learning with ground truth achieved from best known exact solver Concorde (Applegate et al., 2006) for the TSP and unsupervised learning for the VRP. Models are trained using the Adam optimizer for 10 epochs with a batch size of 128 and for reinforcement learning, models are trained for 100 epochs on 128,000 TSP samples which are randomly generated for each epoch (without optimal solutions) with the same batch size and learning rate.

6.2.1.7.2 Evaluation

Model's evaluation id performed on testset, the graph obtained from decoder is converted to valid solution via search strategies described in section 3.5. Optimality gap and predicted average tour length metrics are used to evaluate model performance. Optimality gap is the average percentage ratio of the predicted tour length relative to optimal solution (obtained from exact heuristics such as Concorde, LKH3).

For the TSP, optimal solution is achieved by exact heuristic solvers such as Concorde (Applegate et al., 2006, and LKH3 (Helsgaun, 2017). We also compare our results with Nearest Neighbour which is a non-learned baseline algorithm. We also compare results with OR Tool solver. For the VRP, we consider CVRP and Split Delivery VRP and compare results with results obtained by Nazari et al., (2018) from their RL based solution.

6.2.1.7.3 Results

In this section we report on experimented we conducted using proposed framework where we leveraged exact solvers for optimal solutions as well as learning based solutions to our TSP/VRP in the project.

For the machine learning approaches to TSP and VRP is based on training and evaluating model performance on problem instances of fixed sizes. We train three models training sets of 500k instances with 20, 50 and 100 nodes in each and evaluate them on test dataset of 10k instances. For the training loop, cross-entropy loss with stochastic gradient descent was used and training results in a an adjacency matrix corresponding to a TSP tour.

During the evaluation, adjacency matrix obtained from Graph ConvNet is transformed into a valid solution via search strategies described in previous section. We use predicted tour length and optimality gap (optimal solution is obtained from Concorde solver) following the study (Kool et al., 2019). The average predicted tour length over test dataset is computed as $\frac{1}{n}\sum_{i=1}^{n} l_i^{\wedge}$ and optimality gap is the ratio of predicted tour length relative to the optimal solution over test set, computed as: $\frac{1}{n}\sum_{i=1}^{n} (\frac{l_i}{l_i} - 1)$

Method	N = 20				N=50		N=100			
	Tour length	Opt. gap	Time	Tour length	Opt. gap	Time	Tour length	Opt. gap	Time	
Concord	3.84	0.0%	1m	5.7	0.0%	2m	7.76	0.0%	3m	
LKH3	3.84	0.0%	18s	5.7	0.0%	5m	7.76	0.0%	21m	
Nearest Neighbor	4.5	17.23%	0s	7.0	22.94%	0s	9.68	24.73%	0s	
OR-Tool	3.85	0.37%	-	5.80	1.83%	-	7.99	2.9%	-	
GCN	3.89	1.46%	6s	5.94	4.37%	55s	8.4	8.23	6m	

Table 6.4 Performance of Graph ConvNet against exact and heuristic solutions

The results of Graph ConvNet based model is shown in Table 6.4 where exact solutions are obtained from Concorde and LKH3 solvers. We also show results for the same dataset for the Nearest neighbour heuristic. The GCN model does not perform as well as optimal solution but it achieves results in much less time compared to exact and heuristic solvers. Out of all the models, NN perform worst from tour length prediction perspective but has the best run time.

6.2.2 Routing Protocol Communications Flow

In the Figure 6.7, we show the interaction of routing service with other services in the ICONET project. The routing service is depended on the networking service to get updated PI network as well problem definition which may or may not include vehicle capacities, time-windows during which each location to be service by delivery vehicle. Routing service determines the tour of vehicle and returns a list of nodes as optimal route which is consumed by simulation and shipment service.



Figure 6.7: Integration of routing service with others in ICONET

7 DISCUSSION AND CONCLUSIONS

In this report we analysed the transport processes of shipping, encapsulation, networking and routing under the prism of PI. It is argued that the uniqueness of the PI, arises due to the emphasis on certain aspects of transportation such as multiple parties, more route and warehouse/hub accessibility and multi-leg transport chains. Essentially, this leads to more opportunities for identifying efficiencies through more environmentally friendly routes, bundling, higher potential to reduce empty runs etc.

PI may however increase some of the risks of transport, due for example to an increase in the number of 'touching points' for cargo. The need for process coordination between unknown to each other parties, also places more emphasis on standardisation of the exchanged information. The GPICS data structure was considered as a starting point for developing a comprehensive data structure, that enables the PI to make informed decisions on the utilisation of infrastructure, assets and decentralised capability.

The report presents the Physical Internet (PI) four core services, namely the Shipping, Encapsulation, Routing and Networking. The services have been designed to align with the OLI/ NOLI (and ICONET) layers enabling a standardised approach to the PI implementation. The ontology of PI Links, Nodes and Services is extended as each service is examined into further detail. The services are designed to account for various business types where various use cases arise. The sequence of communications of the PI Services design has been considered for each use case, aiming to enable the development of modular and robust services. The various applications contexts of the PI have also been taking into account, drawing on the adaptations of the four core services to the ICONET Living Lab requirements. In the report we addressed transport processes in the natural sequence in which they occur:

- Shipping: It has an overarching manager role and can be divided into: design; initialization; arrival at PI node; and real time update modules. The function of the first two modules is associated with the request of shipment through the and the development of the PI Order. The Arrival at PI node module handles the sequential hops of PI containers in their route to their destination, while the real time updates module, communications with the IoT platform and collects data to track the performance of the PI shipment against its contractual obligations.
- Encapsulation. It investigates the bin packing algorithm as well as algorithms for overcoming
 its computational complexity. The encapsulation service addresses the encapsulation of cargo
 into PI containers of various sizes and into PI Movers/Means. It offers a generic tool for
 improving operational efficiency and decision making at PI Hubs. Variations of the generic
 model are described that can also be utilised for efficient eCommerce encapsulation.
- Networking: The networking service primary function is network discovery, in order to provide
 a standardised and complete representation of the PI for further decision making. Using the
 GPICS as a guideline, an enhanced data structure is proposed breaking down the PI network
 information into static and dynamic data for PI Links, PI Hubs and PI Movers. Furthermore,
 considering the ICONET Living Labs, several network representation approaches are
 considered focusing at varying network aggregations. A guideline for networking service
 implementation into different contexts is also provided.
- Routing: The routing service investigates the computational complexity and heuristics for improving the solution time for generic and specialised PI routing problems. The performance
of methods utilising an integration of optimisation and machine learning means are discussed in detail, before proposing an PI.

The multiple interconnections of the ICONET Services, enable accommodating into the Physical Internet various types of businesses ranging from Manufacturers, to Logistics operators, and eCommerce. A complex communications system managed by ICONET Shipping Service was shown to handle various types of Use Cases, including:

- the re-routing of shipment when network status has changed,
- the re-assessment of PI Order priority in case of delays, and
- the continuous assessment of shipment metrics against PI Order terms, which in case of violation can lead to re-evaluation of PI Order through Shipper validation.

Queuing theory was discussed for accurately identifying efficient and inefficient routing options, and an enhance PI Link representation was proposed for simplifying and aggregating the PI networks operational features.

Additional analytic components were considered for accommodating industry specific problems, such as the lack of fulfilment origin in the case of eCommerce. Additionally, enhanced PI Node data structure was proposed to accommodate eCommerce relevant information. All the above processes are impacted by the inherent characteristics of transport and logistics which are essentially distributed (in space and in time), multi-party, multi-modal and stochastic.

Information technology can improve many efficiency parameters; however, it cannot eliminate uncertainties and risks that are naturally occurring in transport. PI provides more options to mitigate such risks (e.g. by switching to different modes or routes) thus making transportation potentially more resilient. The proposed information/data structure enables PI Services to make better and smarter operational decision and deliver a more efficient PI.

Services for tracking and tracing cargo through its PI journey may become more important, as in PI cargo may travel through unknown (to the planner) networks. In this report, we illustrated the potential of a new class of route planners that can plan routes through PI following paths across connected π -hubs. These planners go beyond current multimodal transportation planners as the switching of cargo is not only between modes but also between routes crossing different transport networks. This allows to exploit synergies for cargo bundling and more efficient and environmentally friendly routing.

An end-to-end framework for solving large scale routing problems was introduced. The input to the framework is PI network as property graph and the problem instance. The Graph neural network-based encoder-decoder architecture along with search component determine a set of solutions which then is optimized by the policy learning component through supervised learning or reinforcement learning. The supervised learning is based on ground truth – a solution achieved by best known heuristic for a given TSP or VRP problem instance. In our experiments we leveraged LKH3 and Concord solver for TSP and OR-Tool for the VRP. We also experimented with REINFORCE algorithm for learning the policy gradient as baseline for both TSP and VRP. The models trained with supervised learning are dependent on the heuristic search whereas reinforcement learning approaches scale better with more computation as they do not rely on labelled data. Although framework combines the strength heuristic

algorithms with learning over large data using Graph neural networks but it's scope is limited to simple TSP and two variants of the VRP: Capacitated Vehicle Routing Problem (CVRP) and Vehicle Routing problem with Time window (VRPTW).

All Services presented in this report have been integrated with the PoC Platform. The communication between the PoC has been implemented by using either direct (Service X to Service Y) or indirect (Service X – Simulation – Service Y) Application Programming Interfaces (API). The implementation of the PoC is expected to offer further validate the efficiency PI Services deliver and can potentially encourage further investigation.

A barrier in the development of the services has been the lack of visibility to the sources of stochasticity in the logistics supply chain. The gradual improvement of the representation of the ICONET Physical layer, that captures the location and status of services through sensors and data transfers is expected to offer enhanced visibility of supply chain uncertainty, and contribute in overcoming this limitation. As the PI matures, the Core PI Services, will need to handle more functionality and complexity to effectively manage T&L supply chain. Another aspect where further development is anticipated is the communication of PI Services with existing sensing and legacy management systems. As the PI will grow further extensions and special case modules will require to become part of the core ICONET services to allow for increased availability and openness of the network, without compromising existing business operations. This will also encourage the easier expansion of the network with, more reliable data and increased trust among Supply Chain actors.

8 REFERENCES

Ahuja, R., Magnati, T., & Orlin, J., 1993. Network flows: Theory, Algorithms and Applications Prentice Hall.

Applegate, D., Bixby, R., Chvatal, V., & Cook, W. (2006). Concorde TSP solver.

Bello, I., Pham, H., Le, Q. V., Norouzi, M., & Bengio, S. (2016). Neural combinatorial optimization with reinforcement learning. arXiv preprint arXiv:1611.09940.

Bengio, Y., Lodi, A., & Prouvost, A. (2020). Machine learning for combinatorial optimization: a methodological tour d'horizon. European Journal of Operational Research.

Bresson, X., & Laurent, T. (2017). Residual gated graph convnets. arXiv preprint arXiv:1711.07553.

Chopra, S., 2019. Supply Chain Management: strategy, Planning and Operation. Pearson, UK.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., & Dahl, G. E. (2017). Neural message passing for quantum chemistry. arXiv preprint arXiv:1704.01212.

Google. OR-tools, Google optimization tools, 2019. URL https://developers.google.com/optimization

Hamilton, W., Ying, Z. and Leskovec, J., (2017a). Inductive representation learning on large graphs. In Advances in Neural Information Processing Systems (pp. 1024-1034).

Helsgaun, K. (2017). An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems. Roskilde: Roskilde University.

loffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167.

Khalil, E., Dai, H., Zhang, Y., Dilkina, B., & Song, L. (2017). Learning combinatorial optimization algorithms over graphs. In Advances in Neural Information Processing Systems (pp. 6348-6358).

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

Lin, S., & Kernighan, B. W. (1973). An effective heuristic algorithm for the traveling-salesman problem. Operations research, 21(2), 498-516.

Montreuil, E. Ballot, and F. Fontane, An open logistics interconnection model for the Physical Internet. In Proceedings of INCOM 2012 Symposium, Bucharest, Romania. 2012

Nazari, M., Oroojlooy, A., Snyder, L., & Takác, M. (2018). Reinforcement learning for solving the vehicle routing problem. In Advances in Neural Information Processing Systems (pp. 9839-9849).

Pham, T., Tran, T., Phung, D.Q. and Venkatesh, S., 2017, February. Column Networks for Collective Classification. In AAAI (pp. 2485-2491).

Sehnke, F., Osendorfer, C., Rückstieß, T., Graves, A., Peters, J., & Schmidhuber, J. (2010). Parameterexploring policy gradients. Neural Networks, 23(4), 551-559.

Sharraj, R., Ballot, E., Shenle, P., Hakimi, D., Montreuil, B., 2014. Interconnected logistic networks and protocols: simulation-based efficiency assessment. International Journal of Production Research, Vol. 52, No. 11, 3185-3208.

Snelder, M. 2010. Designing robust road networks: a general design method applied to the Netherlands. de Technische Universiteit Delft.

T.N. Kipf and M.Welling. Semi-supervised classification with graph convolutional networks. In ICLR, 2016.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2017). Graph attention networks. arXiv preprint arXiv:1710.10903.

Vinyals, O., Fortunato, M., & Jaitly, N. (2015). Pointer networks. In Advances in neural information processing systems (pp. 2692-2700).

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine learning, 8(3-4), 229-256.