Knowledge-Based Coordination in cyber-physical systems via distributed ledger technologies

Politecnico
di Bari

Department of Electrical and Information Engineering

Electrical and Information Engineering Ph.D. Program
SSD: IINF-05/A – Information Processing Systems

Final dissertation

Knowledge-Based Coordination in Cyber-Physical Systems via Distributed Ledger Technologies

by
Arnaldo Tomasino

Supervisors:
Prof. Michele Ruta
Prof. Floriano Scioscia

Coordinator of Ph.D. Program:
Prof. Mario Carpentieri

37^{th} Cycle, 01/01/2022 – 31/12/2024

La borsa di dottorato è stata cofinanziata con risorse del
Programma Operativo Nazionale Ricerca e Innovazione 2014-2020, risorse FSE REACT-EU
Azione IV.4 "Dottorati e contratti di ricerca su tematiche dell'innovazione"
e Azione IV.5 "Dottorati su tematiche Green"

Politecnico di Bari

Department of Electrical and Information Engineering

Electrical and Information Engineering Ph.D. Program
SSD: IINF-05/A – Information Processing Systems

Final dissertation

Knowledge-Based Coordination in Cyber-Physical Systems via Distributed Ledger Technologies

by
Arnaldo Tomasino

Referees:
Prof. Marco Morana
Prof. Carlo Puliafito

Supervisors:
Prof. Michele Ruta
Prof. Floriano Scioscia

Coordinator of Ph.D. Program:
Prof. Mario Carpentieri

37^{th} Cycle, 01/01/2022 – 31/12/2024

Contents

List of Figures

List of Tables

Acronyms

AA	Abstract Argumentation.
AF	Argumentation Framework.
AI	Artificial Intelligence.
API	Application Programming Interface.
B2B	Business-to-Business.
BFT	Byzantine Fault Tolerance.
BWAF	Bipolar Weighted Argumentation Framework.
CPS	Cyber-Physical System.
DAG	Directed Acyclic Graph.
DAO	Distributed Autonomous Organiza- tions.
DBMS	DataBase Management System.
DL	Description Logic.
DLT	Distributed Ledger Technology.
DLTs	Distributed Ledger Technologies.
EC	Edge Computing.
EPCIS	Electronic Product Code Information Services.
FOL	First Order Logic.
IoE	Internet of Everything.
IoT	Internet of Things.
IoV	Internet of Vehicles.
IPFS	InterPlanetary File System.
IRI	Internationalized Resource Identifier.

KB	Knowledge Base.
KG	Knowledge Graph.
KR	Knowledge Representation.
KRR	Knowledge Representation and Reasoning.
MaaS	Mobility-as-a-Service.
MAS	Multi-Agent System.
MEC	Mobile Edge Computing.
MEM	Micro-Electro-Mechanical System.
NFC	Near-Field Communication.
OWL	Web Ontology Language.
PaaS	Platform-as-a-Service.
PBFT	Practical Byzantine Fault Tolerance.
PoET	Proof of Elapsed Time.
PoW	Proof-of-Work.
RAMI	Reference Architecture Model Industry.
RDF	Resource Description Framework.
ReST	Representational State Transfer.
SA	Structured Argumentation.
SC	Smart Contract.
SIoT	Social Internet of Things.
SLA	Service Level Agreement.
SNS	Social Networking Service.
SOA	Service Oriented Architecture.
SPARQL	SPARQL Protocol and RDF Query Language.
SWoE	Semantic Web of Everything.
SWoT	Semantic Web of Things.

u-KB Ubiquitous Knowledge Base.
URI Uniform Resource Identifier.
UTXO Unspent Transaction Output.

VM Virtual Machine.

W3C World Wide Web Consortium.

Abstract

The physical and digital domains are becoming increasingly intertwined, encompassing not just objects, but also locations, processes and living entities. In this scenario, everyday objects acquire the capability of collecting, storing, processing and exchanging information. This technological trend has been supported by the research on the *Internet of Things* (IoT) and the broader domain of *Internet of Everything* (IoE). Within such context, *Cyber-Physical Systems* (CPSs) have emerged as complex systems that tightly integrate sensing, computation, communication and actuation capabilities in order to control physical processes. Key CPS requirements include responsiveness, reliability and safety.

To cope with the decentralized, heterogeneous and volatile nature of large-scale IoT-based CPSs, this dissertation adopts a *Multi-Agent System* (MAS) perspective, where each device becomes a smart entity endowed with decision-making abilities and adaptable behavior. However, continuous node mobility and intermittent connectivity hinder trust-based interactions among these agents. *Distributed Ledger Technologies* (DLTs) – particularly those built upon *Directed Acyclic Graph* (DAG) structures – provide promising solutions to these coordination challenges in CPS scenarios by ensuring tamper-resistant data exchange without centralized intermediaries.

This dissertation proposes a novel dual-layer *interledger* framework that unifies multiple blockchain platforms through a shared DAG-based backbone. This enables the design of a microservice architecture that enhances scalability by decoupling system components and distributing the computation required for servicing requests, with a DAG-based substrate to enable transparent data management and tracking of events involving multiple different ledgers. Furthermore, investigates methods to integrate *Knowledge Representation and Reasoning* (KRR) for supporting more sophisticated tasks of resource discovery, composition and negotiation by means of *Smart Contracts* (SCs) leveraging automated reasoning. For this purpose,

the *Seesaw* prototype has been implemented demonstrating how semantics-enabled SCs can facilitate robust, logically explainable coordination in complex CPS environments. Moreover, the need to allow more articulate dialogue capabilities among components of pervasive CPSs has motivated the work on a framework based on *Computational Argumentation* theory, in which a graph-based formalism allows the analysis of interrelation among arguments, detecting agreements and conflicts.

As a further step, prototypes in several case studies show how KRR technologies have been integrated into the dual-layer infrastructure to foster scalability, trustworthiness, and intelligence in interconnected cyber-physical domains. In this context, intelligent agents can seamlessly collaborate, engage in dialogue, adapt and evolve to meet the increasingly complex demands of CPS in the Digital Twin era.

Chapter 1

Introduction

The increasing availability of micro- and nano-electronic components for data processing and communication has allowed their integration into everyday objects, endowing them with the capability of collecting, storing, processing, and exchanging information. This transformation has been pushed through the Internet of Things (IoT) vision. In latest years, an evolution from the IoT to the Internet of Everything (IoE) has been occurring, which is further blurring the boundaries between the digital and physical world. In IoE contexts not only objects, but also locations, processes and living entities are more and more interconnected, continuously exchanging data and interacting through attached intelligent micro- and nano-devices. Sharing the same conceptual underpinnings about the integration of information processing elements into physical environments, the Cyber-Physical System (CPS) paradigm refers to a specialization of the IoT and IoE visions characterized by complex systems engineered to oversee physical processes through a close integration of sub-systems for measurement, computation, communications and actuation. CPS principles are deeply concerned with real-time responsiveness, reliability, safety and security. Example applications are expanding in sensitive fields like high-precision manufacturing, telehealth, and intelligent transportation systems.

By architecting CPSs as composed of autonomous interacting agents, Multi-Agent Systems (MASs) can be designed, where each physical component shifts from a mere data source to a smart object enhanced with context-awareness, decision-making capabilities and adaptive behavior. In this way, it is crucial to allow meaningful ways for the agents to interact

with each other and exchange information, ensuring effective, autonomous and intelligent coordination. Furthermore, the nature of interactions among nodes in a CPS is often decentralized, allowing for high flexibility and scalability of the system. In mobile and pervasive computing contexts, however, smart object networks suffer from high volatility of nodes and resources, due to network unreliability, device power limitations, and mobility enabling devices to enter or exit communication range unpredictably. This hinders reliable trust-based coordination, since it is not possible to assume that nodes know and can reach each other all the time.

Distributed Ledger Technologies (DLTs), stemming from the introduction of blockchain with *Bitcoin* [78], have emerged as an interesting perspective in supporting such environments. DLTs enable trustless, tamper-resistant record-keeping and data exchange across distributed networks without the need for central intermediaries. They achieve consensus on transaction validity through various algorithms (*e.g.*, *Proof-of-Work*, *Byzantine Fault Tolerance* derivatives, and Directed Acyclic Graph (DAG)-based *tip selection*), and use cryptographic primitives to ensure authentication, integrity, non-repudiation, and censorship-resistance. However, traditional blockchains, built as linear chains of transaction blocks, suffer from low throughput, high energy consumption and limited scalability, proving inefficient for IoE and real-time use cases. In response, newer DAG data structures have been investigated for DLT infrastructures, streamlining all processes involving the approval and addition of transactions and guaranteeing high-speed, energy-efficient protocols suitable for resource-constrained environments, allowing to bring the advantages of immutability, transparency and security of DLT platforms for IoE-based CPSs. Nevertheless, efficient interoperability and especially scalability still remain open issues in the DLT domain, due to the complexity of protocols and lack of standards.

This Ph.D. thesis addresses the foregoing issues by proposing a dual-layer *inter-ledger* architectural solution that integrates multiple blockchain platforms through a shared DAG-based DLT. The proposed framework realizes a microservice architecture to enhance MAS scalability in CPSs by modularizing and decoupling the system components, allowing independent management and horizontal scaling of blockchain infrastructures, as well as increased interoperability, by means of a public *IOTA Tangle* [94] layer to track interactions and provide a unified interface for seamless data and

service integration across different blockchain platforms.

Yet, as these systems evolve into more sophisticated settings, the demand grows not only for trustless robust data exchange, but also for higher-level capabilities in decision making and adaptive coordination. The integration of Knowledge Representation and Reasoning (KRR) techniques plays an important role in addressing this problem. The emerging Semantic Web of Everything (SWoE) paradigm [106] aims at unifying the Semantic Web with the IoE into a technology stack where machine-understandable knowledge representation and pervasive computing intertwine. In SWoE settings, standard Semantic Web technologies and formalisms (including ontologies, reasoning procedures, and query languages) are embedded into ubiquitous microcontroller-based field devices for sensing and actuation in addition to microprocessor-based edge computing devices. This approach enables automated reasoning over Ubiquitous Knowledge Bases (u-KBs) that encompass semantically annotated data fragments physically scattered across the environment and accessible through collaborative, often decentralized, protocols. In this context, objects should use embedded reasoning engines to integrate fragmented information on the fly into coherent Knowledge Base (KB) materializations, expressed through annotations in standard Semantic Web languages, so as to support further inference tasks. Specifically, this work focuses on KRR-based approaches for resource discovery – where devices reason over available resource descriptions to find the best match – and resource negotiation – where entities "dialogue" with each other to reach a compromise aligned with their requirements – adapting them to support multi-agent coordination in CPSs. This approach leverages Smart Contracts (SCs), *i.e.*, self-executing programs on a distributed ledger automatically enforcing the terms of an agreement among participants, to embed semantic technologies and procedures in the DLT platform.

Such need for articulate "dialogue" capabilities among pervasive intelligent agents in a CPS further motivates the investigation of a structured framework, grounded on *Computational Argumentation* theory, in which fragments of asserted knowledge are represented as atomic information units and a graph-based formalism helps analyzing their interrelations, detecting (partial) agreements and conflicts. Also in this case, to maintain a high level of trust and reliability, functionalities can be integrated into a set

of SCs to obtain transparent, robust and logically explainable outcomes. This thesis explores methodologies that unify these elements to meet the current and future demands of intelligent, interconnected and trustworthy cyber-physical environments.

The remainder of this dissertation is organized as follows:

- **Chapter 2** provides the necessary background, including an overview of Social CPS, the SWoE, and DLTs. It highlights key technological foundations and state-of-the-art developments relevant to this research;

- **Chapter 3** presents the proposed inter-ledger architecture for multi-agent CPS coordination. The design of a dual-layer DLT-based microservice architecture is discussed, emphasizing scalability, modularity, and suitability for IoT and CPS environments. Furthermore, this chapter also includes a proposal to illustrate how SCs can enable semantic-based services for resource discovery, negotiation, and composition;

- **Chapter 4** describes the argumentative deductive framework outlined for CPS coordination. It elaborates on the *Bipolar Weighted Argumentation Framework* (BWAF) for argumentation graph construction, propagation-based ranking semantics for argument acceptability evaluation, and the associated algorithms;

- **Chapter 5** explores practical applications and case studies, developed to validate the proposed methods and frameworks. Scenarios include public traceability of confidential processes in Business-to-Business (B2B) marketplaces, industrial CPS event tracking, and semantic-enhanced blockchain-based smart mobility services;

- **Chapter 6** concludes the dissertation by summarizing the main contributions and outlining future research perspectives, emphasizing open challenges and potential advancements in knowledge-based CPS coordination leveraging DLTs.

Chapter 2

Background

This chapter provides an overview of (Social) CPSs, the SWoE, and DLTs, outlining the state of the art and open issues. Technological background and motivation are clarified for the investigations reported in this Ph.D. dissertation.

2.1 Social Cyber-Physical Systems

In the IoT paradigm, everyday objects are augmented with communication and computation capabilities. Progress in Micro-Electro-Mechanical System (MEM) miniaturization and integration are facilitating the convergence of the physical and digital worlds, where subsystems for physical measurement and actuation work together with embedded information processing and communication subsystems. This has led to the rise of systems known as Cyber-Physical System, consisting in a large number of sensors, actuators, and control devices, connected by a network and sharing common application-level goals. Such goals are achieved by coordinating tasks concerning the acquisition, processing and analysis of data – along with context information – and the exploitation of results to act back onto the physical environment.

In this type of architectures, autonomous dynamic coordination is a pivotal requirement due to the increasing complexity and numerousness of individual elements. This calls for *smart objects* that are able to understand the context they are inserted in and take decisions dynamically based on manifold factors, including the state of surrounding objects and places.

Oftentimes, CPSs are also socio-technical systems, where human-in-the-loop processes are observed and controlled: in such cases, smart devices have to take into account users' state, activities and profiles as well. Thus, CPS are increasingly designed as Social MASs where each smart device in a given environment acts as an autonomous agent and meaningful agent relationships should be established automatically in order to support flexible orchestration and choreography patterns as well as human-computer interaction modes.

Research in the so-called Social Internet of Things (SIoT) [5] is exploring models and architectures to reach this goal. Interaction paradigms are often borrowed from Social Networking Services (SNSs) for human users, properly adapted to the peculiarities and requirements of MASs. In order to face the intrinsic unpredictability of pervasive computing contexts, interactions in IoT-based CPSs are structured according to a *social model* for object information interchange. SIoT can offer several benefits for CPSs in terms of interoperability, autonomicity, versatility and coordination. However, to achieve really cohesive CPSs endowed with versatile cooperation and dynamic choreography of components, connected smart objects should be able to represent, discover and share information and services described in a high-level, articulate way by means of machine-understandable formalisms. Semantic Web technologies are candidates for such a role, as they can grant interoperability grounded on logic-based formal semantics [123].

2.1.1 Requirements

Digital societies call for increasingly intelligent, distributed and flexible service and resource management. As cities evolve toward a living organism model on both physical and digital planes [127], pervasive computing infrastructures based on IoT sensors and smart devices act as its nervous system. While early architectures strongly relied on Cloud Computing infrastructures, the Edge Computing (EC) paradigm has emerged due to the adoption of capable devices enabling several key data management and processing tasks to be performed at the edge of the network. This allows to achieve shorter response latencies, more efficient bandwidth usage and higher data privacy and security. Advanced IoT scenarios require highly

flexible composition and dynamic reconfiguration of services, resources and devices, each endowed with well defined roles and responsibilities. Due to the broad heterogeneity of vendors, device classes, as well as application areas, interoperability and decentralized autonomous decisions are still challenging issues. In the latest years, a few interesting novel paradigms have been proposed, which may be applied at the EC level in IoT architectures. They typically exploit Artificial Intelligence (AI) to simulate aspects of human decision-making behaviors [118] in networked multi-agent systems.

The SIoT paradigm [5] introduces a layer of social networking principles into the IoT ecosystem to enhance the efficiency, usability, and intelligence of interconnected devices. The main requirements of SIoT architectures include:

- dynamic interactions: smart objects are designed to interact not only based on predefined technical protocols, but also through social relationships, akin to human social networks. Relationships mirror human social connections, such as co-location, co-work, friendship, and ownership, and facilitate more intuitive interactions independent from user engagement and from human SNS;

- context-aware collaboration: IoT devices act as autonomous agents building networks of social relationships and exploiting them to share information and services, and perform tasks collaboratively [76]. The social layer adds context and relational data to the interactions, enabling devices to make more informed and context-aware decisions;

- resource optimization: by leveraging social relationships, smart devices can optimize the allocation and negotiate access to shared resources, like network bandwidth or energy power, based on their social standing or priority within the network.

SIoT approaches have found significant applications in real-world scenarios, including smart mobility [35], smart building [112], and smart manufacturing [90]. Nevertheless, ensuring interoperability among diverse devices from different manufacturers remains a critical challenge for the seamless functioning of SIoT networks. Monitoring and maintaining social relationships among a large number of devices in a trustworthy and reliable way is a challenging task, requiring sophisticated algorithms and frameworks.

2.1.2 State of the art

In latest years, SNSs have changed personal interaction habits and relationships management on a global scale. SNS members create personal profiles with basic information about themselves; connect with other users in either bidirectional (*e.g.*, *friendship*, *group*) or unidirectional (*e.g.*, *follower*) relationships; post text and/or multimedia items on their *wall* (*i.e.*, log) for sharing with their contacts; flag (*tag*) some contacts to associate them and draw their attention to certain content elements; respond to content published by other users with comments and reactions (*e.g.*, *like*). SNS adopters generally manifest an intention to continue using them [63], because SNSs provide both *extrinsic* – *i.e.*, usefulness – and *intrinsic* – *i.e.*, gratification – value. Their utility also grows with a *network effect* as they connect more users, and particularly *complementary* ones [63], since opportunities increase for discovering information, resources and services.

Similarly, in the SIoT [5], objects engage with one another independently from direct user interactions. Things act as autonomous agents, building networks of social relationships and exploiting them to share information and services more effectively. Research on SIoT is very active and covers a wide range of related topics, as recent surveys [124, 4] show.

Several classifications of relationship types exist in literature. In [7] and subsequent works, *parental* (same manufacturer), *co-location* (same environment), *co-work* (cooperation), *co-ownership* (same owner) and *social* (sporadic or frequent contact) are identified as basic object relationships. The analysis in [132] is inspired by literature on human SNS, instead, and results in an ontology allowing objects to manage their policies, friends and reputation. The ontology model developed in [58] includes social relationships among events, people and objects in IoT environments.

The SIoT aims at extended device collaboration. Service Oriented Architecture (SOA) is one of the dominant paradigms, where cooperation is mediated by the production, provisioning and usage of *services*. In this model, intelligent service discovery is a core capability. Most SOA-based SIoT approaches rank services w.r.t. a request/profile specification based on a combination of (i) object social connectivity, (ii) object mobility patterns, and (iii) preference similarity. For instance, in [61] a cosine similarity is computed combining metrics (ii) and (iii), improving results w.r.t. each

single method, although preferences and resources are described by means of a simplistic set of keywords, lacking formal meaning.

Semantics-based approaches improve both context-awareness and discovery capabilities in SOAs. In [101] distributed KB management supports machine-to-machine social interactions in control networks. Every connected object proactively discovers other nodes in the network; requesters are then able to distribute queries automatically among known devices equipped with reasoning engines. Unfortunately, supported inferences are very basic, limiting the applicability of the proposal. In [45] semantics-based situation detection and goal retrieval are exploited for matchmaking with task profiles for recommending activities to users, based on their current context. Nevertheless, social interactions occur only between devices and users. Furthermore, adopted rule-based reasoning could not retrieve approximate matches when exact ones do not exist. Likewise, [44] focuses on activities of daily living in smart homes, generating task-oriented recommendations for user's situational goals. In [62] Near-Field Communication (NFC) technology mediates social interactions between everyday objects and agents running on mobile devices. Ontology-based representations support context-awareness, enabling both reactive and proactive agent behaviors, exploiting rule-based reasoning for planning toward situation-dependent goals. Though interesting, the approach does not appear general enough to support a wide range of SIoT scenarios.

Recent SIoT proposals increasingly adopt cloud-assisted architectures, where social objects are *virtualized*, *i.e.*, functionalities are mapped to software agents managed by Platform-as-a-Service (PaaS) infrastructures. Relevant examples include *Paraimpu* [93], *Lysis* [38], *Smartbuddy* [91] and the platforms in [44], [136] and [57]. Notably, the *ASSIST* framework [53] exploits semantics-based rules to implement social network primitives. While granting higher scalability, availability and robustness, cloud-assisted approaches can exhibit non-negligible drawbacks, including longer delays, higher network load and greater energy usage for pervasive devices [36]. As such, it seems appropriate essentially for scenarios and applications where a dependable global networking infrastructure is available. Although SIoT research trends take this kind of connectivity more and more for granted, many IoT technologies in the present and near future cannot provide it [6].

Among challenging scenarios, the SIoT paradigm is being increasingly

applied to *Vehicular Ad-hoc NETworks* (VANETs) and urban mobility [125]. The fast motion of nodes (*i.e.*, vehicles) leads to highly volatile network topologies as well as strict computation and communication latency constraints for safety-related services. Furthermore, situation awareness in such environments requires information fusion from multiple heterogeneous sources. Social interaction paradigms in VANETs –often denominated *Social Internet of Vehicles* (SIoV) [1]– have been demonstrated to be effective in improving information discovery [109] and resource allocation [143].

Trust management is one of the most relevant issues in the SIoT [141]. Trust metrics can be retrieved from friends (recommendation), reliable intermediaries (reputation) or direct knowledge [29]. In [81] both a *subjective* and an *objective* (*i.e.*, shared) model are introduced, exploiting techniques mutuated from peer-to-peer networks. The collaborative Cognitive Radio framework in [82] leverages the SIoT relationship types described above in [7] to improve both the reach of channel status queries and the trustworthiness of information through weights depending on the degree of friendship. A refinement is proposed in [25], where trust is defined as a function of reputation on past transactions, relationship type and energy status. Reputation combines direct and friend recommendation scores, while the relationship type between two devices is derived from a correlation of trustworthiness w.r.t. mutual friends. Similarly, in [49] a *honesty* model is composed out of spatial (object proximity), temporal (frequency and duration of interactions), relationship and credibility (cooperativeness and penalties) metrics. An interesting approach is adopted in [23], albeit in that case honesty referred to the absence of malicious behaviors: the main goal is, in fact, to devise a robust trust model against attacks to service discovery and management. For this purpose, a trust propagation and aggregation scheme is introduced, conceptually similar to the objective trust model in [81]. The above works, however, aim only at improving transaction success and do not consider dynamic social relationships. That issue is tackled in [87], exploiting Bayesian belief propagation networks.

As a final remark, all the above works focus either on service discovery and composition or on dynamic trust management, disregarding or trivially simulating the other aspect. In [115] a framework integrates semantics-based SOA and dynamic relationship management in SIoT for CPS, formalizing and analyzing mutual influences between these two aspects.

2.2 The Semantic Web of Everything

The ongoing miniaturization of IoT devices, combined with the enhanced integration of individuals, places, processes, and data through pervasive mobile networks, is driving the evolution of the IoT into the IoE. Within the IoE paradigm, streams of data are generated by living entities, objects, locations, and phenomenons of interest. This data proliferation requires sophisticated information management strategies, robust security measures and scalable trust mechanisms. In parallel with the evolution of the IoT, its integration with *Semantic Web* technologies and tools has undergone substantial advancements, progressing from the Semantic Web of Things (SWoT) to the more recent frameworks for the SWoE.

The Semantic Web initiative, a collaborative movement led by the World Wide Web Consortium (W3C), aims to transform the *World Wide Web* into a universal platform for data exchange, where machines can interpret the meaning of information and use it as knowledge to take decisions and execute tasks. This objective is achieved through a suite of technologies and standards, including the Resource Description Framework (RDF) [121] for structured metadata representation, the Web Ontology Language (OWL) [88] for the creation and sharing of *ontologies* as structured domain vocabularies, and SPARQL Protocol and RDF Query Language (SPARQL) [96] for querying RDF datasets and Knowledge Graphs (KGs). By augmenting Web resources with semantic metadata, the Semantic Web embeds *machine-interpretable* information within the existing Web infrastructure. This enhancement facilitates advanced automated processes and applications – such as intelligent search engines, sophisticated data integration, and personalized user agents – by enabling machines to comprehend and respond to complex human queries based on a semantic understanding of digital information. Fundamental to this vision are the disciplines of Knowledge Representation (KR) and Automated Reasoning, underpinned by RDF and OWL to express statements along with their context and relationships. Automated reasoning empowers software agents to perform logical deductions for undertaking complex tasks, thereby supporting intelligent search and informed decision-making capabilities. While the SWoT aims to integrate the Semantic Web with IoT devices, the SWoE vision extends the scope to networks of micro- and nano-devices disseminated

in physical locations and closely tied to objects, people and processes. This integration fosters the creation of intelligent and autonomous environments wherein objects operate as smart agents. These agents are capable of processing and analyzing contextual data from their surroundings, autonomously communicating and exchanging information to ultimately fulfill their mission and goals.

2.2.1 Knowledge Representation for the Semantic Web of Everything

The W3C standards for modeling ontologies in the Semantic Web are formally based on Description Logics (DLs) [8], a class of knowledge representation languages that provide essential semantics to express knowledge and support inference tasks. DLs are decidable fragments of First Order Logic (FOL) [19, 30] and comprise three core components:

- *concepts*, denoting sets of objects within a specific domain;
- *individuals*, the actual objects in the domain, representing instances of concepts;
- *roles*, defining binary relationships between pairs of concepts.

By employing various *constructors*, these elements can be combined to form complex expressions. Their formal semantics is defined via an *interpretation* $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ where Δ represents the universe of discourse (the domain), and $\cdot^{\mathcal{I}}$ is the interpretation function that maps each term to a subset of Δ. In this semantic framework, the conjunction of concepts is interpreted as the intersection of sets: $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$. The disjunction of concepts corresponds to the union of sets: $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$. When included, the negation operator $\neg$ signifies the complement of a set.

An *ontology* in the context of DLs – also known as a *terminology*, terminological box, or *TBox* [126] – is composed of assertions involving concepts and roles. There are two primary types of assertions:

- *inclusion assertions* ($A \sqsubseteq D$), which define hierarchical "is-a" relationships between concepts, indicating that concept A is a specialization of concept D;

- *equivalence assertions* ($A \equiv D$), used to state that two expressions represent the same set of instances or to assign names to complex expressions.

The collection of individuals and the relationships between them form what is known as the *assertion box* (*ABox*). A KB is given by a ⟨TBox, ABox⟩ pair.

DLs are primarily distinguished by the constructors they provide. The *Attributive Language* ($\mathcal{AL}$), introduced in [120], serves as a minimal yet practically significant language within this family. The constructs available in $\mathcal{AL}$ are as follows:

- $\top$, *universal concept*: represents all objects within the domain;
- $\bot$, *bottom concept*: corresponds to the empty set;
- A, *atomic concepts*: denotes all objects that are instances of the concept A;
- $\neg A$, *atomic negation*: includes all objects not belonging to the concept A;
- $C \sqcap D$, *intersection*: the set of objects that are instances of both concepts C and D;
- $\forall R.C$, *universal restriction*: consists of objects x such that for every object y, if x is related to y via relation R, then y is an instance of concept C;
- $\exists R$, *unqualified existential restriction*: objects that are related to at least one object through relation R.

To enhance its expressive capabilities, the $\mathcal{AL}$ language can be extended with additional constructs. The notation $\mathcal{AL}$ $[\mathcal{N}\ ][\mathcal{U}][\mathcal{E}][\mathcal{C}\ ][\mathcal{I}]$ is used to indicate popular $\mathcal{AL}$ language extensions. In this work, the *Attributive Language with unqualified Number restrictions* ($\mathcal{ALN}$) is adopted as reference DL. With respect to $\mathcal{AL}$, it includes *unqualified number restrictions* $\geq nR$, $\leq nR$, $= nR$, denoting objects that are related to at least, at most,

or exactly n objects via relation R, respectively[1]. Table 2.1 provides a summary of the syntax and semantics of $\mathcal{ALN}$ constructors and assertions.

Table 2.1: Syntax and semantics of $\mathcal{ALN}$

Name	**Syntax**	**Semantics**
Top	$\top$	$\Delta^{\mathcal{I}}$
Bottom	$\bot$	$\emptyset$
Intersection	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
Atomic negation	$\neg A$	$\Delta^{\mathcal{I}} \backslash A^{\mathcal{I}}$
Universal quantification	$\forall R.C$	$\{d_1 \mid \forall d_2 : (d_1, d_2) \in R^{\mathcal{I}} \rightarrow \ d_2 \in C^{\mathcal{I}}\}$
Number restrictions	$\geq nR$	$\{d_1 \mid \sharp\{d_2 \mid (d_1, d_2) \in R^{\mathcal{I}}\} \geq n\}$
	$\leq nR$	$\{d_1 \mid \sharp\{d_2 \mid (d_1, d_2) \in R^{\mathcal{I}}\} \leq n\}$
Inclusion	$A \sqsubseteq D$	$A^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
Equivalence	$A \equiv D$	$A^{\mathcal{I}} = D^{\mathcal{I}}$

In practical implementations, a standardized syntax is essential for storing and exchanging ontology documents across various tools and applications. Ontologies can be serialized using any RDF triple-based syntax, such as *RDF/XML* [121] , which encapsulates RDF within the eXtensible Markup Language, or *Turtle* [97]. Additional OWL-specific syntaxes like *Functional* [88] and *OWL/XML* [75] encode OWL axioms directly, without translating them into RDF triples. Lastly, the *Manchester syntax* [41] is designed to be more accessible to users without a formal background in logic.

Web Ontology Language (OWL)

In 2009 and late 2012, the W3C released official Recommendations for two iterations of OWL. OWL 2 expanded upon the foundational principles established in the initial version by enhancing the language expressiveness and extending support for datatypes and annotations.

Entities serve as the fundamental building blocks of OWL 2 ontologies, forming the basis for defining an ontology through named terms. Each entity is uniquely identified by an Internationalized Resource Identifier (IRI) [31], which is an extension of the Uniform Resource Identifier (URI) [14] that allows a broader set of characters.

[1]It is useful to notice that $\exists R$ is equivalent to $\geq 1R$ and that $= nR$ is a shortcut for $\geq nR \sqcap \leq nR$.

Entities are categorized as follows:

- **Classes**: groups of individuals, which can form hierarchies where one class is a subclass of another.
- **Individuals**: specific objects or instances that belong to classes.
- **Object Properties**: relationships that link individuals to one another.
- **Datatype Properties**: attributes that assign data values to individuals.
- **Annotation Properties**: used to provide metadata about the ontology itself rather than the domain of interest, such as information about the author of a specific axiom or the date it was added.

OWL also supports the definition of *anonymous individuals*, which are useful for representing information about an entity without specifying a unique identifier.

Entities can be combined into more complex *expressions* using logical *constructors*. These expressions define sets of individuals by specifying criteria related to their properties. Individuals that meet these criteria are considered instances of the corresponding class expressions. An OWL 2 ontology comprises a collection of *axioms* – basic statements that assert truths within the domain of interest by combining entities and expressions. OWL supports a variety of axiom types, enabling the composition of entities and expressions into multiple forms of logical assertions.

The extensive range of logical constructors supported by OWL contributes to its high degree of expressiveness. To manage this complexity and cater to specific, industrially relevant use cases, such as improved reasoning scalability and efficient query answering, OWL 2 introduced several *profiles* for the language:

- *OWL 2 EL*: optimized for ontologies with large numbers of properties or classes, making it particularly suitable for fields like life sciences;
- *OWL 2 QL*: designed for applications that require efficient access to large volumes of instance data, facilitating seamless integration with relational databases;

- *OWL 2 RL*: aimed at applications that need scalable reasoning capabilities without substantially compromising expressiveness.

Each profile strikes a balance between expressiveness and performance, allowing practitioners to choose the most appropriate subset for their specific use case, ensuring that OWL remains versatile and applicable across a wide range of domains. Nonetheless, knowledge-based systems are not mandated to fully conform to any specific OWL profile: the supported OWL constructs can be arbitrarily restricted to align with DLs that are known to have favorable computational complexity and practical performance characteristics, while still being sufficiently expressive for useful applications.

2.2.2 Ubiquitous Knowledge Bases

Addressing the challenges inherent in dynamic and decentralized SWoE environments requires efficient methods for information storage, management, and discovery, alongside transparent access to local information sources. The concept of u-KB, originally introduced in [108], serves as a pivotal architectural model for integrating the Semantic Web with the IoE, thereby actualizing the SWoE. By adopting Semantic Web languages and technologies, the u-KB facilitates detailed annotation and categorization of resources. This enables semantic-based applications to utilize tools for discovery, querying and reasoning, all grounded in formal logic principles established by the Semantic Web. Such semantic enrichment is essential for rich *machine-to-machine* application and service platforms and for supporting the automation levels required in dynamic CPS contexts.

Central to the u-KB framework is its ability to allow physical objects, equipped with semantic capabilities, to be discovered, queried, and inventoried collaboratively in a peer-to-peer manner without the need for centralized control or coordination. This is achieved through a comprehensive architectural blueprint that bridges mobile ad-hoc networks used in ubiquitous computing with the Internet. The u-KB incorporates a distributed application-layer protocol operating on a peer-to-peer basis, which facilitates the dissemination and discovery of knowledge. Information is gathered using various identification and sensing technologies and is subsequently employed by inference engines and semantics-aware applications

via a uniform set of operations in both pervasive environments and the Web.

A significant innovation is the adoption of a *content-centric* networking approach. Rather than targeting the physical locations of data storage, this technique focuses on the content itself, aligning with the volatile nature of the IoE where entities are frequently mobile or transient. This paradigm shift is crucial for network adaptability, ensuring that content can be retrieved regardless of the changing states or positions of devices. The u-KB is fully decentralized and utilizes a two-level infrastructure comprising a *field layer* for connecting embedded micro-devices and a *discovery layer* for inter-host communication. Implementing suitable peer-to-peer protocols enhances the system's decentralization, resilience, and scalability, enabling devices to autonomously discover and query information without reliance on a centralized authority. Given those requirements and advantages, one of the architectural frameworks that better adapt to this approach is blockchain, due to its decentralized peer-to-peer nature and transparency in the storage of data. Aligning with this approach, Section 3.2 outlines an infrastructure where multiple blockchains are grouped and managed by a single lightweight DLT layer to coordinate their interactions. Scattering semantic annotated information on these blockchains can be considered as a step forward towards a physical substrate implementation of a u-KB approach, as proposed in Section 3.4.1.

Embedded within the u-KB are query primitives and reasoning tools that facilitate sophisticated resource discovery by means of semantic matchmaking, allowing the system to transcend mere data retrieval. User queries and device data are interpreted within specific contexts, ensuring that the information most relevant to the user's current needs is identified and gathered. A distinctive feature of the framework is its support for device autonomy and distributed collaboration. Through shared vocabularies and ontologies, devices participating in a u-KB can independently register, deregister, and recognize other devices and services. This ensures effective communication by sharing and interpreting information in a unified language, thereby promoting seamless interoperability among devices.

As outlined in Figure 2.1, the u-KB framework adopts a multi-layer architecture designed to facilitate semantic information dissemination and resource discovery in pervasive environments. The interaction among its

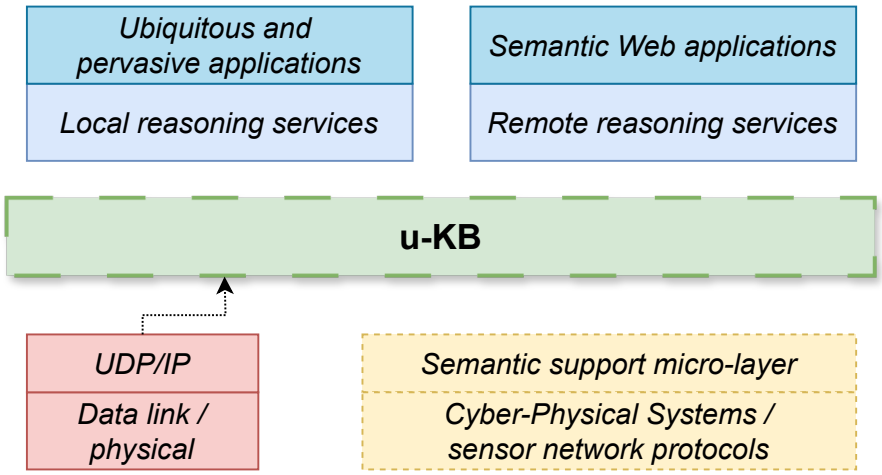


Figure 2.1: Ubiquitous Knowledge Base framework.

layers orchestrates the flow of information from the physical world to the application layer, enabling intelligent data processing and decision-making. The architecture consists of the following layers:

- **Data Link/Physical Layer:** includes *field layer* protocols and technologies that enable the physical interconnection of embedded devices in the environment. Examples include RFID [22], Bluetooth [17], ZigBee [27], KNX [55], among others. A *discovery layer* within this tier manages the identification of resources and services, allowing each network host to act as a cluster head for field devices within its immediate range by utilizing available communication interfaces;

- **Semantic Support Micro-Layer:** conceptually conceived as the core of the framework, this layer ensures interoperability with mobile ad-hoc networks of embedded devices and sensors. It adapts mobile identification and sensing technologies from the physical layer to meet the semantic requirements of the framework, effectively bridging the gap between raw sensor data and semantic processing needs;

- **u-KB Layer:** provides unified access to information from semantics-enhanced devices and sensors populating a smart environment. It employs the *Internet Protocol* (IP) for fundamental addressing and routing within local networks and between autonomous networks, in-

cluding wide area networks and the Internet, thereby ensuring seamless communication across different network scales;

- **Reasoning services:** offered to both local pervasive applications and remote Semantic Web applications, these services enable logic-based querying and reasoning capabilities essential for processing semantically annotated information. They facilitate sophisticated inference and support explainability of responses.

By coordinating the functionalities of these layers, the u-KB framework effectively manages the flow of semantic information from physical devices to high-level applications, supporting the advanced features required in pervasive computing environments.

2.2.3 Automated Reasoning for the Semantic Web of Everything

Due to their close association with DLs, OWL ontologies are able to represent complex domain knowledge, supporting automated reasoning processes that derive new insights from explicitly defined facts and relationships.

Focusing on the $\mathcal{ALN}$ DL, structural inference algorithms are well-established for both standard inferences ([8]) and non-standard, non-monotonic inferences [105]. These algorithms rely on preprocessing steps involving concept unfolding and normalization into *Conjunctive Normal Form* (CNF). *Concept unfolding* recursively expands the terminological axioms in the TBox within concept expressions, thereby eliminating the need for the TBox in subsequent inference procedures. Subsequent to unfolding, *CNF normalization* converts the expanded concept expressions into a canonical form that preserves their original semantics. In the context of $\mathcal{ALN}$ CNF, every concept expression is either the bottom concept ($\bot$) or a conjunction ($\sqcap$) of the following components: (i) atomic concepts, which may be negated; (ii) number restrictions involving greater-than ($\geq$) and less-than ($\leq$) constraints, limited to one per role; and (iii) universal restrictions ($\forall$), also limited to one per role, with their fillers recursively expressed in CNF. Given a DL ontology $\mathcal{T}$ and S, R two concepts in $\mathcal{T}$, the *satisfiability* and *subsumption* standard inference services provided by DL-based systems [8] can be formalized as follows:

- **satisfiability**: checks if S can have instances w.r.t. the ontology $\mathcal{T}$, *i.e.*, $\mathcal{T} \not\models S \sqsubseteq \bot$. An unsatisfiable class contains a contradiction, which implies that it cannot have any instance;

- **subsumption**: checks if S is more specific than R w.r.t. the ontology $\mathcal{T}$, *i.e.*, $\mathcal{T} \models S \sqsubseteq R$. In this case, all instances of S are also instances of R.

After $\mathcal{ALN}$ concept expressions have been unfolded and normalized, subsumption and satisfiability can be determined by examining their structure. This involves treating the expressions as sets of primitive atoms representing the $\mathcal{ALN}$ constructs detailed in Table 2.1.

Although the standard inference services discussed earlier are valuable for traditional semantic-enabled applications, they often fall short in scenarios that require more than Boolean answers, which are prevalent in the SWoT and the SWoE. In such cases, non-standard, non-monotonic inferences [105] become more relevant, as they provide explanations for inference outcomes and allow for the retraction of conflicting information under the *Open World Assumption* (OWA). This is particularly important in SWoE scenarios, which are characterized by highly volatile data and fragmented information [107, 142]. For instance, in [114], features of samples were annotated using fragments of conjunctive concept expressions, while target classes were represented with concept expressions, both referred to a shared ontology. Then, the process of *semantic matchmaking* is employed to identify the most relevant elements within a set of registered *resources* in response to a given *request*, where both the request and resources are represented as satisfiable concept expressions with respect to a common set of axioms $\mathcal{T}$ in a DL. Differently from standard inference services, the outcome of semantic matchmaking algorithms is a ranked list of concepts, each accompanied by a score that indicates its semantic relevance to the submitted query. This can facilitate not only resource discovery in SWoE environments, but also advanced procedures involving negotiation and composition, as well as support data stream analytics by transforming statistical classification problems into semantic matchmaking ones. In this dissertation, these techniques are exploited to enhance resource and service discovery paradigms and enable complex interactions for negotiation procedures tailored for CPS environments (Section 3.4).

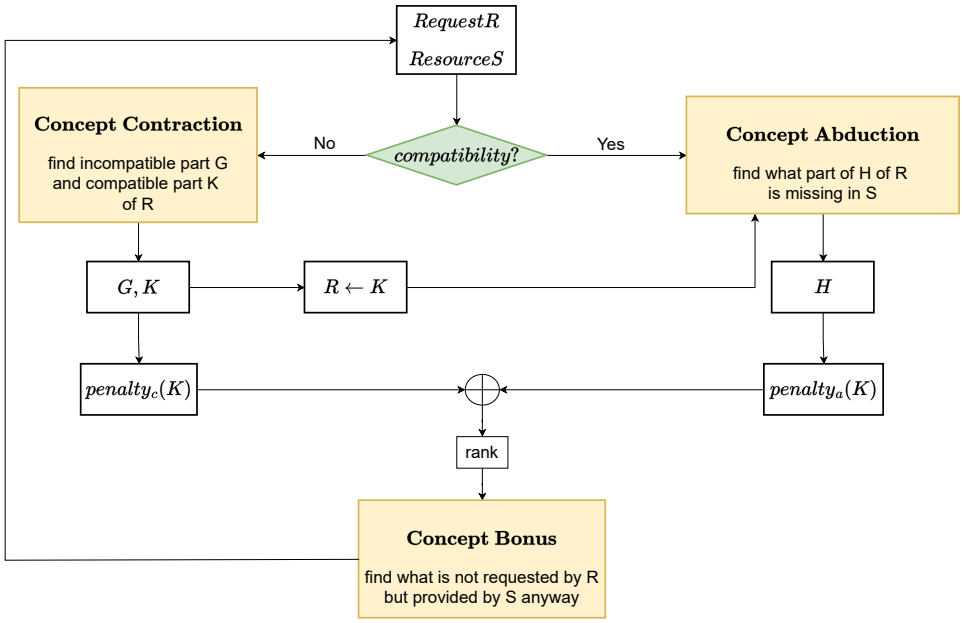


Figure 2.2: Semantic matchmaking framework.

The *semantic matchmaking* reference framework, illustrated in Figure 2.2, integrates both standard and non-standard inference services to effectively match requests with resources. For clarity, let us consider a set of axioms $\mathcal{T}$ in $\mathcal{ALN}$, and two concepts R and S – representing a *request* and a *resource*, respectively – both satisfiable w.r.t. $\mathcal{T}$.

The matchmaking process begins with a preliminary **Consistency check** to determine whether the conjunction of R and S is satisfiable in $\mathcal{T}$. Formally, this involves verifying that $\mathcal{T} \models R \sqcap S \not\sqsubseteq \bot$. If this check fails, conflicting requirements are detected as the resource only partially matches the request. To resolve this, ***Concept Contraction*** (CC) is employed, a non-standard inference service that identifies and removes the conflicting parts of the request. CC computes a pair of concepts $\langle G, K \rangle$ such that $\mathcal{T} \models R \equiv G \sqcap K$ and $\mathcal{T} \not\models K \sqcap S \sqsubseteq \bot$. Here, G (for "Give up") represents the conflicting requirements to be retracted from R, while K (for "Keep") is the remainder of the request that is consistent with S. Thus, the solution G provides an explanation for the inconsistency between R and S, enabling a transition from a partial match to a potential match by considering the contracted request K.

On the other hand, if $R \sqcap S$ is satisfiable but the resource S does not

fully satisfy the request R – *i.e.*,, $\mathcal{T} \not\models S \sqsubseteq R$ – a potential match occurs. This is the case where the request provides more information than the resource. To analyze this, the ***Concept Abduction*** (CA) non-standard inference service is used. CA computes a concept H such that $\mathcal{T} \models S \sqcap H \sqsubseteq R$ and $S \sqcap H$ is satisfiable in $\mathcal{T}$. The concept H (standing for "Hypothesis") represents the additional requirements from R that are not specified in S. This provides insight into what is missing in S for it to fully satisfy R, facilitating the progression from a potential match to a full match, also known as a subsume match [59]. A full match occurs when $\mathcal{T} \models S \sqsubseteq R$, indicating that the resource meets all the features specified in the request.

Additionally, ***Concept Bonus*** (CB) is another valuable inference service in matchmaking contexts. A resource S may offer features not explicitly requested in R, perhaps because the requester was unaware of them or did not consider them important. CB extracts a *bonus* concept B from S, capturing these additional features. This information can be leveraged to refine the query, possibly leading to a more satisfactory matching outcome.

It is important to note that these inference services operate under the OWA [105]: the absence of some information in a concept expression is interpreted not as negation but as an unspecified constraint, possibly due to that information being unknown or deemed irrelevant.

The CNF representation of concept expressions in $\mathcal{ALN}$ induces a metric space equipped with a *norm* operator $\|\cdot\|$. Within the previously discussed matchmaking framework, the CNF norms of the concepts G and H serve as semantic distance *penalties* for CC and CA, respectively. These penalties are used to rank a set of resources with respect to a given request. Similarly, the norm $\|B\|$ provides a relevance measure for the *Bonus* concept obtained through CB. In [105] penalty values have been proposed where:

1. each (possibly negated) atomic concept counts as 1;
2. for each role, number restrictions are weighted as the relative difference between cardinality values in R and S w.r.t. the cardinality value in R (if R lacks a number restriction on that role, the penalty is 1);
3. for each value restriction, the penalty is computed recursively on the filler as above.

Algorithms for CA, CB, and CC in [106] are designed to satisfy a minimality criterion, as the goal is typically to hypothesize or give up as little information as possible.

While CA, CB, CC, and CD are effective in one-to-one discovery, matchmaking, and negotiation scenarios, the non-standard inference service known as ***Concept Covering*** (CCov) can be utilized to assemble a collection of elementary expressions to address complex requests [122]. Formally, given a concept expression R (the request) and a set of concept expressions $\mathcal{S} = \{S_1, S_2, ..., S_n\}$ representing available resources in a u-KB, where both R and each S_i are satisfiable with respect to the reference ontology $\mathcal{T}$, the output of CCov is a pair $\langle \mathcal{S}_c, H \rangle$. Here, $\mathcal{S}_c \subseteq \mathcal{S}$ comprises concepts that collectively form a (potentially incomplete) covering of R with respect to $\mathcal{T}$, and H denotes the portion of R that is not covered by the elements in S_c. Essentially, CCov processes a set of resources alongside a request with the aim to:

- cover the request: satisfy the features specified in the request R as comprehensively as possible by combining resources;
- explain uncovered parts: provide an explanation for any aspects of R that remain unsatisfied.

2.3 Distributed Ledger Technologies

Current research on DLTs started with the introduction of the open-source *Bitcoin* electronic currency platform and the *blockchain* data structures and protocols underpinning it [78]. Despite the controversies often surrounding cryptocurrencies and their uses, blockchain and subsequent DLTs are intrinsically useful technologies for the implementation of robust decentralized databases and transactional systems, attested by successful adoption in a wide range of applications.

2.3.1 Blockchain basics

Blockchain combines a data structure and protocol for *trustless* distributed transactional systems. It works as a distributed ledger, recording transactions occurred in a given time span. Transactions are grouped in *blocks*,

whose size depends on the particular blockchain system. Each block contains a hash value in the header section, computed on both the contents of the block and the hash of the previous block, as illustrated in Figure 2.3. This cryptographic technique helps in forming an immutable chain of blocks, preventing tampering with old blocks without consensus among the nodes.

In traditional distributed databases, the properties of irreversibility (i.e., no committed transaction can be reverted or altered) and censorship prevention (i.e., all valid transactions have the same probability to be stored) are guaranteed by a trusted intermediary. Instead, blockchain systems avoid intermediaries by including a set of nodes into a peer-to-peer network: they approve transactions through a distributed *consensus* protocol. This decentralized model makes blockchain a *trustless* collaboration platform: participants are not required to trust any central authority or individual external actor; instead, they rely on mechanisms based on cryptography, consensus protocols and decentralized verification processes to ensure data integrity. This way, it is guaranteed that no single node or small group of colluding nodes can force the addition, removal or modification of data. The minimum percentage of colluding nodes among participating nodes depends on the particular consensus mechanism. Moreover, the adopted protocol itself guarantees authentication (via private/public keys) and non-repudiation, thereby eliminating the need for mutual trust among participants.

In Bitcoin, the blockchain is employed as a shared ledger to store currency transfer transactions, approved by the participating nodes with the Proof-of-Work (PoW) consensus algorithm [133]. Since the success of Bitcoin, many other blockchain-based electronic currency platforms have been introduced. *Ethereum*[2] is perhaps the most popular open blockchain platform after Bitcoin: it manages the exchange of a cryptocurrency, known as *Ether*, which is used to pay for services and transaction submission on the Ethereum network. Additionally, Ethereum is the pioneering platform designed to deploy and execute decentralized applications that run Smart Contracts (SCs), *i.e.*, programs encoding and enforcing cooperative processes among two or more parties. Consensus about SCs in a blockchain is reached through a parallel execution on every peer in the network that

[2]Ethereum Project: `https://www.ethereum.org/`

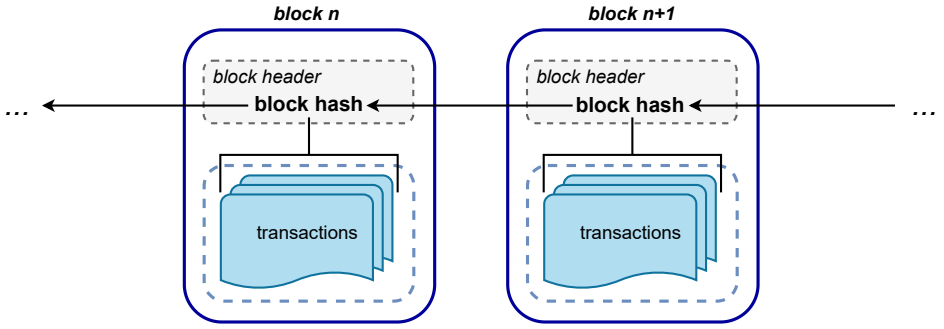


Figure 2.3: Blockchain structure

deploys them, whereas the original concept of SC [128] required a trusted mediator, restraining a large adoption of the approach. This effectively makes every SC-enabled blockchains a general-purpose application platform based on a distributed replicated Virtual Machine (VM) and a secure distributed database.

Blockchain systems have been experiencing an increasing diversification of architectures, technologies and applications, leading to the development of the DLT research area. Useful surveys on that and particularly its integration in the IoT are treated in [104, 86, 138]. The *Hyperledger* initiative[3], guided by the Linux Foundation, incubates several blockchain-related open-source technologies aimed at industrial applications. The first blockchain infrastructure developed under the *Hyperledger* umbrella project was *Fabric*[4], a private and permissioned blockchain, characterized by a modular architecture and a strong authentication protocol. Another example is the former *Sawtooth*, now maintained by the Splinter community[5], which enables permissionless as well as permissioned blockchains in heterogeneous computing contexts. Both these blockchain platforms support the development and execution of SC.

In addition to ledgers organized as linear chains of blocks of transactions, models based on Directed Acyclic Graph (DAG) data structures have been increasingly proposed in latest years, with different structures for the ledger of transactions. Most DAG DLTs have been designed to enable higher performance and scalability w.r.t. traditional blockchain platforms.

[3]Hyperledger: `https://www.hyperledger.org/`
[4]Hyperledger Fabric: `https://www.hyperledger.org/projects/fabric`
[5]Sawtooth (Splinter): https://github.com/splintercommunity/sawtooth-core/

A notable example is *IOTA*[6], a DAG DLT solution tailored for the IoT industry, based on the *Tangle* data structure [94]. In the Tangle, transactions are appended to the tips of the DAG and each new transaction must validate a minimum of two previous transactions according to a *tip selection* algorithm. Nodes contribute to security by approving and verifying new transactions, thereby avoiding potential conflicts.

The main key design decisions for blockchain systems are as follows:

- *Network access policy.* In *permissioned* blockchains, only nodes in a specific list are admitted. In *permissionless* ones, any node can join at any time.

- *Decentralization level.* A blockchain network is *public* if any node can join the consensus protocol, even anonymously. It is known as *private*, instead, if only a set of uniquely identified nodes in an allow-list are admitted to participate in the consensus. A hybrid model is *consortium* blockchains, where only a subset of all nodes, among those who use the platform, are admitted to validation through consensus. Blockchain design is deeply influenced by this choice. In fact, in public blockchain platforms, participants need to be rewarded for their computational effort. On the contrary, private platforms are more suited for contexts where collaboration needs to be more controlled and access itself is a reward, as it allows users to interact with partners and exploit available services and resources.

- *Transaction model.* In a blockchain, an *asset* represents a digital token or unit that can be registered and exchanged, and whose ownership can be tracked immutably. At any time, each node typically owns some assets in a certain quantity. There exist two main models to represent assets on a blockchain:

  - in the Unspent Transaction Output (UTXO) model, a transfer from A to B is modeled as *consuming* (*i.e.*, deleting) records for A's spent assets and producing (*i.e.*, adding) new ones for B's received assets. The UTXO model is similar to a bank statement of account; it allows simpler reconstruction of current state from

[6]IOTA: (`https://www.iota.org/`)

a transaction log and is typically adopted by electronic currency systems;

  - in the *account-based* model, every node has an account reporting all its assets, which is updated by transactions. The account-based model is better suited for general purpose applications, making it the only practical choice for SC-based blockchains. However, it can make transaction processing slower.

- *Consensus.* Requirements for the consensus algorithm vary from permissionless to permissioned platforms. The former require stricter consensus methods to guarantee data security from malicious nodes colluding to subvert the blockchain. For instance, Bitcoin introduced PoW, based on solving a cryptographic challenge to allow a node to submit its block to the others and then add it into the chain. The computational burden required by PoW requires incentivization of consensus participants, by allowing nodes to *mine* fresh currency when their candidate blocks are selected. On the other hand, in permissioned platforms each node is accountable, so consensus constraints may be relaxed in order to reduce the computational load. Examples of such algorithms include Byzantine Fault Tolerance (BFT) [133] derivatives. The more lightweight Proof of Elapsed Time (PoET) consensus, based on a leader-election lottery system, can be adopted if CPU-level enforcement of atomic execution of critical code sections is available. A comparison of blockchain consensus protocols is in [86]. In DAG DLTs, consensus algorithms differ from the ones for blockchains because there is no concept of *miner* or leader-based algorithms [92]. In fact, user themselves are responsible for the ordering of transactions. Traditional algorithms like PoW are mainly used as anti-spam protection. In IOTA, the decision about which of the tip transaction will be selected and approved by a new incoming transaction is guided by the *tip selection* algorithm: it is based on a *random walk* starting from old and confirmed transactions, guided by weights assigned to all transactions in the Tangle [94].

- *SC language.* SC can adopt any specification and execution formalism, such as procedural (imperative) languages or logical (declarative) languages or automata [46]. The majority of current proposal adopt

computationally complete programming languages, either existing (*e.g.*, C++, Java, Rust) or created for the purpose (*e.g.*, Ethereum's *Solidity*).

2.3.2 State of the art

Numerous blockchain-based initiatives, with varying stages of maturity, have been proposed in different domains. In fact, blockchain technologies are facilitating the development of transparent, trustless peer-to-peer models for the advancement of reliable IoT applications [98]. In this context, there is an observable trend towards the development of a versatile blockchain-based middleware layer designed for application-agnostic machine-to-machine interactions. This innovation is geared towards enabling IoT service marketplaces to operate with little to no human intervention [26].

Among the most explored use cases, intelligent supply chain, smart cities, and information-centric marketplaces [86, 104] can be mentioned. Smart mobility utilizing 5G vehicular networks stands out as a particularly interesting application area, encompassing data, services and energy exchange [137, 65, 52]. Smart industry is another prominent blockchain application area, due to its standards-based approach and economic impact [24]. In this scenario, relevant uses include asset tracking and supply chain management, where blockchain can contribute with its widely recognized benefits of trustless Distributed Autonomous Organiza- tions (DAO) collaboration [56, 70, 71]. Furthermore, blockchain can also contribute to enhance existing industry standards, such as in the case of Electronic Product Code Information Services (EPCIS) [60]. These approaches range from simple transactional ledgers for asset transfer, granting high throughput at negligible cost [26], to more sophisticated system based on SCs. The latter allows any application logic to be implemented and embedded in the blockchain [26], including discoverable, composable and verifiable multi-step business processes in multi-party SOA [83]. The *Reference Architecture Model Industry (RAMI) 4.0* [47] specification outlines a SOA designed to support cross-organizational interoperability and cooperation throughout the full lifecycle of objects and processes in industrial CPS. It effectively leverages existing advancements in service discovery and com-

position, like those demonstrated by case studies [16, 110].

However, the wider adoption of traditional blockchain has highlighted some intrinsic disadvantages of this technology mainly in terms of performance and scalability. PoW and SCs still have a significant impact in terms of transaction throughput [26]. A rise in transaction volume during a specific period adversely impacts throughput, as the platform does not have enough time to handle all transactions efficiently. Moreover, the benefits of smart contracts come at a not-negligible cost in terms of concurrent execution of transactions and, consequently, system throughput [26]. The inability of nodes to predict the computational demands of executing SC, potentially triggering other Smart Contracts, complicates concurrent execution of all transactions within a block. This contrasts with blockchain systems like Bitcoin that facilitate asset transfers, where transaction semantics are fixed and their dependencies and ordering conditions are predetermined. Nevertheless, this intrinsic rigidity of Bitcoin systems have always hindered the potential for becoming a general-purpose solution. These issues pose a substantial challenge for implementing sophisticated blockchain-based applications in the CPS domain, which demands both high-speed and low-latency transaction processing capabilities. Research is very active in this direction, with many proposals aimed at enhancing blockchain scalability. Interesting surveys on the state of art of blockchain scalability are [119, 144, 102]. One of the explored approaches relies on offloading PoW computation to nearby edge computing nodes, such as in the Mobile Edge Computing (MEC) architecture proposed in [66]. Others propose the optimization of consensus protocols [133] and the introduction of parallelism in a blockchain through *sidechains* and/or *sharding* [28]. The latter is a parallelization technique borrowed from Database Management Systems, consisting in splitting data elements (*e.g.*, rows in relational databases) horizontally across node subsets in a cluster [135]. Sidechain approaches, instead, work by supporting the main chain with auxiliary chains.

DAG-based DLTs enable different proposals to enhance blockchain scalability. The DAG DLT data structure allows a parallel and asynchronous processing of transactions. This reduces latency and increases throughput, producing shorter confirmation time. Moreover, in most DAG DLT platforms, consensus protocols are much simpler and there is no concept of

mining. The most notable example is IOTA, built specifically for the IoT and, thus, optimized for high scalability, low resource consumption, and secure data exchange. A thorough analysis of performance, security and robustness properties of the IOTA platform in IoT is conducted in [139]. First of all, traditional blockchain platforms are deemed as unsuitable for IoT applications and, in general, scenarios requiring fast response time, due to their low throughput, resource intensive mining algorithms and also costly transaction fees. For this purpose, the conducted analysis of IOTA performance focuses on three key metrics: (i) throughput, *i.e.*, the number of transactions the network can handle per second, (ii) inherent latency, *i.e.*, the time from transaction initiation to its attachment to the Tangle, and (iii) confirmation latency, *i.e.*, the time until a transaction is confirmed. Results showed that the major performance issues involve communication among nodes for synchronization, which impacts their CPU load. Moreover, the identified bottleneck is not the tip selection process for transaction validation but the database queries required during transaction initiation. Specifically, IOTA's method of checking for unique address usage when creating transactions involves scanning all transactions in the database, which becomes increasingly time-consuming as the database grows, leading to degraded performance.

DAG DLT infrastructures have also been adopted in scenarios like smart grid [95] and smart city [145]. An interesting application of IOTA for *Internet of Drones* networks is proposed in [72]. Here, a substratum based on IOTA is used to manage identities of drones, ground control stations and control rooms and also to keep track of all transactions involving the tasks of surveillance, delivery and inspection. The experimental campaign compared the deployment of the same system on both Ethereum and IOTA: results demonstrate how deploying such a system on IOTA produces a much faster and more energy efficient solution. In [74] a comparison of traditional blockchain, DAG DLT and *Holochain* (`https://www.holochain.org/`) is carried out through a case study based on energy trading: the work highlights how DAGs can effectively address scalability issues, but at the expense of greater vulnerability to attacks and less decentralization in sensitive applications.

In many proposed applications, a DAG DLT middleware layer is used for the exchange of data generated by individuals through their personal

devices. In data marketplaces, IoT devices have the role of producers of data that can be consumed by other parties interacting through a DLT layer. Applications in [95, 42] are based on the IOTA framework and related protocols, such as *IOTA Streams* for securing message streams and *IOTA Wallet* for the exchange of tokens. This enables the implementation of data marketplaces where information can also be protected against fraud or manipulation and greater privacy can be guaranteed. However, in the majority of state-of-the-art proposals, IoT devices are considered merely as sources of data to be shared in the distributed ledger, rather than agents with the ability to engage in complex interactions and coordinate autonomously by leveraging a DLT platform. The main purpose of using DLTs in these scenarios lies in its advantages of data protection, immutability and transparency at the same time. This work aims to outline an architecture able to show how the DLT substratum can play an active role in coordinating agents in an IoT-based CPS.

Logic-based technologies represent a growing perspective for the integration and coordination of IoT devices and blockchain technologies in business applications [43]. In [130], a blockchain-based framework mediates robot coalition formation employing SCs, where both robot sensors/actuators and environmental parameters are exposed as resources annotated w.r.t. an ontology. Ontology-based SC design in [54] is used for traceability purposes in supply chains, while the ontology in [34] is proposed for annotating transactions to facilitate searching for blockchain contents by semantic-enabled user agents. Other existing logical frameworks have been suggested for SC specification and execution, such as *Defeasible Reasoning* in [46] and *Linear Temporal Logic* in [43], which are already employed widely for the formalization of legal contracts and for model checking, respectively. In [110] the first semantic-based resource discovery approach for blockchain systems is proposed.

Chapter 3

Interledger architecture for Cyber-Physical System coordination

Interledger architectures have been introduced in latest years to interface different DLT platforms in a hierarchical system, in order to exploit the peculiarities and benefits of each adopted DLT while minimizing their limitations. This chapter describes a novel proposal for a dual-layer interledger architecture to enable scalable and robust coordination of autonomous IoT-based smart agents in distributed CPSs. As a notable feature, KRR languages and methods recalled in Chapter 2 provide the algorithmic foundation for flexible SOA support including resource discovery, negotiation, and composition.

3.1 Motivation

In recent years, several DLT architectures have been proposed to address scalability limitations in traditional blockchain platforms and enhance interoperability in IoT and CPS contexts.

In [50], the authors introduce a dual-layer infrastructure where a substratum of IoT devices and sensors are directly associated each with a sidechain, while a consortium blockchain layer interconnects the sidechains through *notary nodes*. The sidechains track sensor data and events in the corresponding device network, while the consortium blockchain maintains

the log of all events concerning the interconnected IoT networks. In many applications, notary nodes play a relevant role to ensure interoperability between different blockchains. In this context, they act as trusted intermediaries that facilitate the communication and validation of transactions between different blockchain networks, ensuring that transferred assets and information are handled securely and consistently. In [50], notary nodes act as gateways, enabling cross-chain transactions that involve micropayments for IoT data and relying on decentralized file systems such as InterPlanetary File System (IPFS) as storage mechanism. In [40], a multi-chain blockchain infrastructure employing notary schemes is introduced. Here, a layer of *sub-chains* is tied to IoT devices to collect data, and each sub-chain is linked to a *main chain* through dedicated notary modules. IoT devices do not work as blockchain nodes, but they communicate with their respective sub-chain through dedicated gateways that invoke SCs.

While both the above approaches leverage a multi-layer DLT infrastructure, their main purpose is facilitating data sharing among IoT devices, rather than providing services for device coordination in complex architectures, like CPSs. Thus, resource-constrained devices are treated only as data sources for exchange or collection, rather than active agents. Furthermore, even if the proposed approaches mention SC invocation, they lack properly rigorous methods for managing the semantics of exchanged information.

The architectural model outlined in [10] organizes the DLT-based infrastructure into four layers. *Layer-1* includes a public blockchain infrastructure to ensure seamless interoperability among different blockchain platforms employed on *Layer-2*. On *Layer-2* each platform is local to a group of IoT devices to gather their data and guarantee restricted access to sensitive information. Interoperability among blockchains is enabled by a common data structure to provide transfer of assets and, in general, information. *Layer-3* and *Layer-4* concern IoT device network access and user authorization, respectively, to make up a comprehensive security strategy.

Besides notary nodes acting as gateways, many proposed solutions for the interoperability among DLT platforms are based on the establishment of a shared standard for transaction exchange. An approach of this kind is in [51], where a unified transaction format is defined to enable the packing and unpacking of transactions between two communicating blockchains.

With the goal of balancing throughput and number of nodes in blockchain networks for IoT applications, the architecture proposed in [84] adopts a hierarchical structure: a central, high-throughput blockchain network, referred to as *Core Engine*, functions as the backbone, with a limited number of nodes utilizing a BFT-variant consensus, whereas multiple sub-blockchains are deployed to handle the scalability demands of high numbers of IoT devices. These sub-blockchains delegate specific tasks from the Core Engine, *i.e.* economic transactions with the *Payment sub-engine*, SC execution with the *Compute sub-engine* and storage management with the *Storage sub-engine*. Notary nodes govern the Core Engine and collaborate with subordinate *Agent* nodes – ranging from servers to edge and IoT devices – to manage the sub-engines. IoT applications are deployed on notary nodes, which provide access to sub-engine functionalities through an Application Programming Interface (API) and handle requests to enable parallel processing and reduce the storage requirements for individual nodes, thus facilitating broader participation in the network. Although this architecture presents a valuable solution for a more scalable and service-oriented architecture in IoT scenarios, it relies on fixed sub-modules each specialized for a limited set of tasks, which restricts its potential for more generalized and flexible interactions.

In [99] a two-tier blockchain architecture is proposed for IoT data recording, particularly acting on the consensus mechanism. In this system, blockchain nodes are subdivided in two levels, according to their different roles they play. IoT data is initially acquired and processed by near low-tier clusters of *orderer* nodes, and then another set of *validator* nodes approves and stores the batches and blocks in the blockchain of the single top-tier level. This work mainly addresses scalability issues by minimizing processing delays and organizing computation and data storing in geographically distributed clusters. Moreover, the authors highlight how the leader-based Practical Byzantine Fault Tolerance (PBFT) consensus mechanism adds complexity that hinders the system scalability, although still proposing it as a solution to the computational burden of the classic PoW consensus protocol. Instead of relying on a single centralized leader, the network organizes consensus through a set of multiple entry points to initiate a consensus round, with bandwidth reservation. This demonstrate how the consensus algorithms pose a serious challenge when addressing

interoperability and especially scalability issues in blockchain platforms; thus, it is important to carefully chose optimized and faster consensus algorithms when working with traditional blockchains, or opt for newer and more streamlined DLT platforms.

This work proposes a DLT-based two-layer microservice architecture to address scalability and interoperability challenges in the DLT scenario. Traditional blockchain platforms are grouped in a layer called *L2*, ensuring that each transaction is tracked on these local, privately managed platforms. At the same time, a public layer named *L1*, based on the lightweight *IOTA* DAG DLT, keeps track of processing steps executed in *L2* blockchains and can accept external requests. The proposed architecture aims to offer several key advantages:

- *modularity and decoupling*: every layer includes components with a well-defined role, and is designed to function as a stand-alone infrastructure, as explained in greater detail in the next section. In fact, the *L2* blockchains platforms and the *L1* IOTA layer are always able to operate independently to one another. Unlike the multi-layer DLT-based architectures analyzed above, there are no tightly coupled components, *e.g. sidechains* directly managing IoT data, *sub-chains* linked through notaries or, in general, components binding a main chain to different other chains and SCs whose execution and data depend on SCs on other chains. The proposed modular, decoupled design ensures that the system remains flexible, scalable, and adaptable to various blockchain technologies and services;

- *scalability*: the design of the system simplifies the addition of new blockchain infrastructures to layer *L2* in a way that is transparent to both users and nodes utilizing the IOTA protocol. This means that new data and services can be included seamlessly. For horizontal scaling of blockchain platforms, minimal changes to the overall system are required, *e.g.*, only *L2* components need to be properly notified of the updates, leaving *L1* unaffected, as specified in the next section. This design can reduce the storage and computational load on each individual blockchain platform in *L2*;

- *suitability for IoT deployment*: while other architectures rely on traditional blockchain platforms, the proposed system utilizes the *IOTA*

Tangle to track all request-response interactions. The *IOTA Tangle* is well-suited for IoT environments due to its lightweight design and fee-less transactions, making it more appropriate for the constrained resources of IoT devices. A potential deployment scenario could involve edge devices managing the *L1* IOTA infrastructure, while offloading the requested computations to (on premise, cloud or hybrid) blockchain networks on *L2*. These features make the architecture suitable for deployment in CPS scenarios, enabling intelligent coordination of devices not only by means of shared data, but also exchanging useful services in a transparent and secure manner. As outlined below, SOA primitives which can be built upon the proposed architecture include device and service registration, device and service discovery, service negotiation, and more.

The distribution of data and services on the traditional blockchain platforms in *L2* provides a further benefit, as it allows each one of them to be independently managed by a single organization. For instance, in Smart City scenarios, each blockchain could manage data or services whose domain is related to specific urban areas. This division of competences and responsibilities can be applied in the field of *supply chain* management as well. In such cases, each group of *L2* blockchains can be managed by a specific consortium of enterprises and maintain data or provide services related to their specific industry sector. By coordinating all private blockchains within a network supported by the IOTA Tangle, it becomes possible to track requests involving multiple entities and groups. The system presents a unified interface to external users, while internally it manages data, services, and infrastructure in a specialized manner.

An additional advantage of this approach is the ability to easily support semantic-based services. The core of the semantic layer in the proposed infrastructure consists in knowledge bases distributed across several blockchains, as opposing to more traditional approaches that concentrate them on a single platform. This allows the realization of an interledger-backed u-KB infrastructure, where the model described in Section 2.2.2 is realized by means of on-the-fly retrieval of resources from various autonomous blockchains in response to a request.

3.2 DLT-based microservice architecture

In the proposed architectural framework, users interact with the system as if it were a *black box*, submitting service requests and receiving corresponding responses, when required. Under the hood, the processing is executed in a distributed manner, involving multiple blockchains interconnected via a substrate based on the IOTA Tangle. This design ensures that every processing step is recorded on the respective blockchains, while the IOTA Tangle keeps track of all request-response interactions.

The overall architecture can be divided into two distinct layers, as shown in Figure 3.1:

- **Layer 1 (L1)** (lower part in Figure 3.1, with blue background): this layer comprises components that interact with the end users on one side and a shared *message broker* on the other side, while keeping track of the overall interaction flow into the IOTA Tangle;
- **Layer 2 (L2)** (upper part in Figure 3.1, with red background): this layer consists of a component called *gateway* that interacts with the shared message broker and communicates with the various independent blockchain networks to coordinate distributed computation through the *manager*.

In what follows, a detailed description of each component is provided.

IOTA Client. It is the *L1* component, which manages user interactions. The IOTA Client provides a set of endpoints that allow users to send requests, which are tracked and forwarded to the appropriate components capable of processing them. At this stage, the received user message is not analyzed to interpret the details of the specific invocation. Instead, the IOTA Client implements logic to package and store data into the IOTA Tangle and to submit the request to the shared Message Broker (described below). As a result, the IOTA Client is designed to work as a completely agnostic entity with respect to both the specific service request and the underlying *L2* blockchain providing that service.

Furthermore, since IOTA has always been deemed as a protocol well-suited for IoT, this client supports implementation as a lightweight process on an IoT device, which sends requests and receives responses by interacting with IOTA for invoking the specific associated services. In this way, several

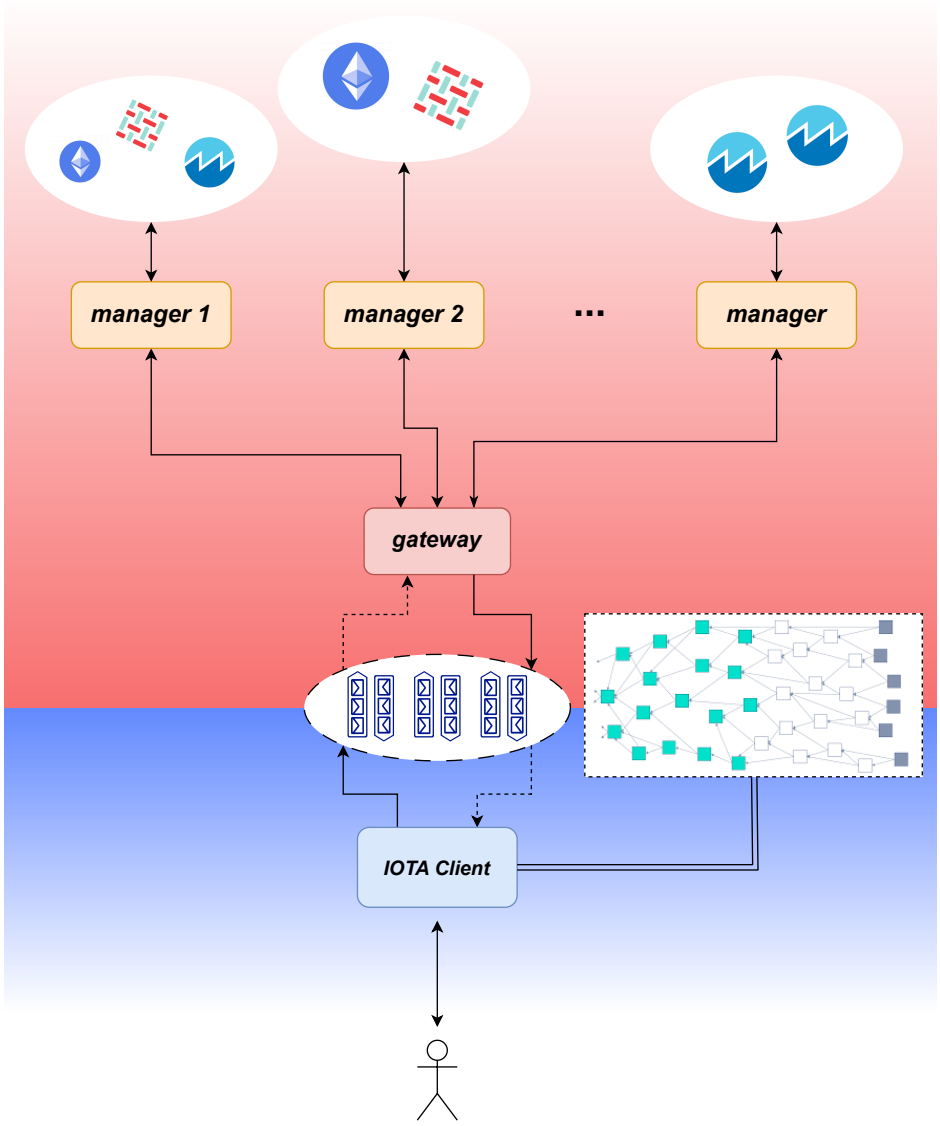


Figure 3.1: Proposed interledger architecture

IoT devices can interact as autonomous agents in a complex CPS, where all interactions are service-oriented and mediated by the dual-layer DLT infrastructure.

Gateway. In the *L2* layer, a Gateway is responsible for managing the interfacing logic between the shared Message Broker and the networks of the available private blockchain platforms. This involves collecting and interpreting requests from the Message Broker and subsequently routing them to the appropriate set of blockchain platforms.

The Gateway provides an endpoint through which a *Manager* can register itself as service provider within the system. The complete process is illustrated in Figure 3.2. The Manager submits a *registration message* that includes the following fields:

- **networkID**: identifier of the connected blockchain network;
- **networkURL**: address of the specific *Manager* component, enabling the gateway to locate it when receiving a request message;
- **services**: list of services provided as SCs on the blockchain. For each service, the following details are specified:
  - list of required parameters to be included in the service invocation request;
  - structure of the response, if provided.

The Gateway interprets registration messages, updates the list of available services in a dedicated storage, and creates a corresponding topic on the shared Message Broker. The topic is named according to the *networkID*. The Gateway then subscribes to this topic, allowing the reception of incoming messages: this mechanism ensures that the Gateway can properly receive requests for specific services originating from IOTA Clients. In this way, the services catalog that the connected blockchains intend to expose becomes accessible publicly. It can be noticed that the process exclusively involves components within layer *L2*, which is the only layer with knowledge of the details regarding the available services.

The IOTA Client exposes a Representational State Transfer (ReST) API endpoint to allow external users to obtain an overview of the services registered with the associated Gateway component. For this purpose, the

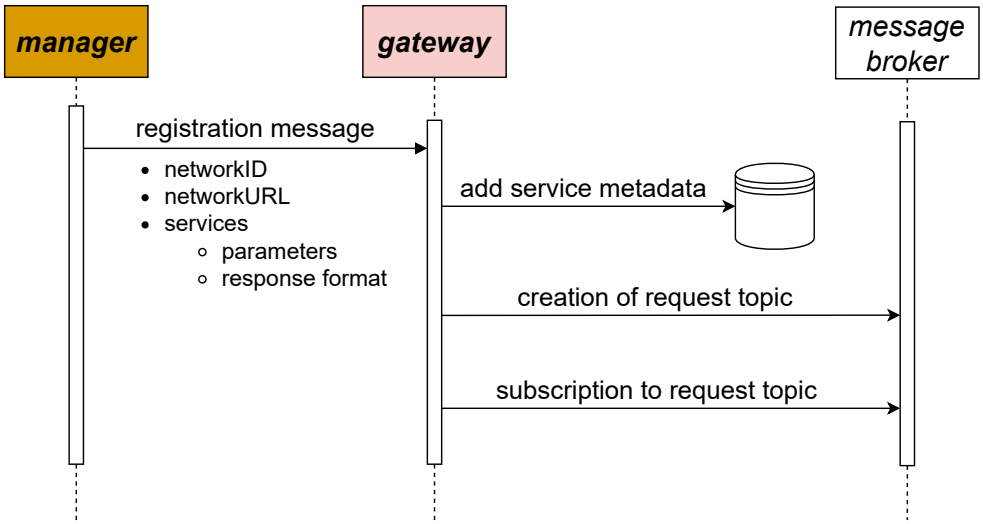


Figure 3.2: Service registration process

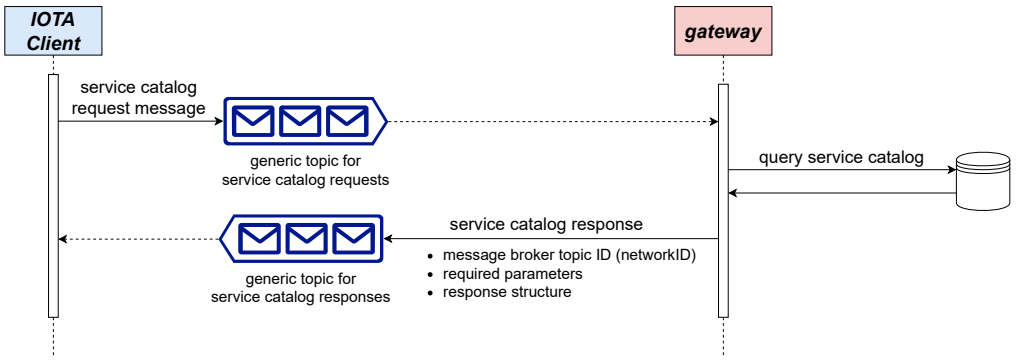


Figure 3.3: Service query process

IOTA Client initializes two general topics that are not linked to any specific Manager. The first topic is dedicated to collecting request messages, while the other handles response messages. As illustrated in Figure 3.3, when a user sends a request to retrieve all available services, the IOTA Client component publishes the message to the general topic. Since the Gateway has already subscribed to it, it retrieves the request and generates a response that contains the information extracted from its private database, updated during the registration phase of the Manager components. Specifically, this response includes a comprehensive list of all services offered by the *L2* network. For each service, it specifies:

- the message broker topic, which corresponds to the registered *networkID*;
- a description of the required parameters for the service request;
- the structure of the potential response.

The Gateway posts the response to the general topic, where it is retrieved by the IOTA Client, which forwards it to the user. As usual, this interaction is mediated by the Message Broker: the decision of leveraging the Message Broker and the Gateway for this interaction is driven by the goal of maintaining the *L1* layer agnostic to the specifications of the registered services.

Message Broker. The Message Broker acts as a shared component, enabling bidirectional, event-driven communication between the two levels of the proposed architecture via the publish-subscribe paradigm. It manages a set of topic pairs: each pair is associated with a network of blockchain platforms that intend to expose specific services or public data. Within the Message Broker, these topics are linked to individual Manager components in the infrastructure and are organized as follows:

- one topic receives requests from users interacting with the IOTA Client. It is created during the service registration phase when a Manager registers to a Gateway, which, in turn, subscribes to the topic;
- the other topic stores the responses generated by the Manager components, which are inserted into this topic by the Gateway. When the

IOTA Client receives a request specifying a given topic, it first checks if the corresponding response topic already exist. If it does not, it is created for the first time. Its name is directly derived from the corresponding request topic name (*e.g.*,, by appending a `_response` suffix).

Manager The required services are executed involving procedures implemented within Smart Contracts (SCs). Each *L2* blockchain network is linked to a Manager component that enables bidirectional communication with the Gateway. This component allows the Gateway to send service requests and it transmits back responses, containing data produced internally in the managed *L2* blockchains. While the Manager's implementation is tailored to the specific blockchain platforms it can support, it still remains decoupled from the remaining components. Furthermore, given the Manager's oversight of all connected blockchain platforms for executing computations related to a particular SC, service availability can be improved by distributing the computational load across these platforms using load-balancing strategies. In this scenario, all the blockchain platforms connected to the same Manager can make available the implementation of the same SC, allowing the Manager to distribute the load accordingly to their availability.

Although integrated into the proposed two-level DLT platform, the Manager is designed to also work independently, outside the complete infrastructure. In general, when a SC is invoked using the Manager via one of its endpoints, a response may be expected. If the request was forwarded by the Gateway, the corresponding response is published to the relevant topic on the shared Message Broker, as explained above. However, even if the execution of the SC is purely local on a *L2* blockchain platform, without interledger distribution, the Manager provides the benefit of a continued tracking of the overall flow of service requests and responses on the IOTA Tangle at *L1*.

3.3 Public DAG-based integration layer

The IOTA Client component works as the main entry point of the infrastructure, and it is responsible to manage the connection to the IOTA

Tangle. It is implemented as a server, configured to expose a set of endpoints through a ReST API. The primary purpose of this API is to allow users to submit requests for services executed on connected distributed blockchains. All requests and responses are tracked on the Tangle through the IOTA platform substrate.

Additionally, the IOTA Client establishes a connection with the shared Message Broker. This connection is employed to:

- publish requests messages on a given topic;
- subscribe to topics where responses will be posted.

When a user sends a request, the topic name is extracted from the request message content, and the request is published to that topic. At the same time, the IOTA Client subscribes to the associated response topic – creating it if it does not already exist – in order to be able to receive incoming response messages. This setup allows the IOTA Client to operate asynchronously: the only task it executes is publishing requests and process responses as soon as they are produced on the corresponding topic. As a result, multiple messages can be handled concurrently, improving overall responsiveness.

For additional privacy guarantees, messages are stored in isolated channels on the Tangle, managed through the *IOTA Streams*[1] framework, designed for secure message transmission on the Tangle via a specific transport layer. It offers the possibility to track messages on private channels, where all users that wish to publish messages on said type of channel must be authorized first (except for the channel *author*), after going through the subscription process illustrated in Figure 3.4. In this context, two types of users can be defined: ***authors*** and ***subscribers***. First of all, the author generates a private key that identifies its instance in the IOTA framework and must be kept secret. The author is in charge of initializing a channel: this process produces an *announcement message* that is shared off-chain to all other agents that wish to interact on the created channel, or it may be stored on the author's running instance so that it can be explicitly requested by potential subscribers. In turn, a subscriber uses the announcement message to connect to the channel by sending a *subscription*

[1] `https://github.com/iotaledger/streams`

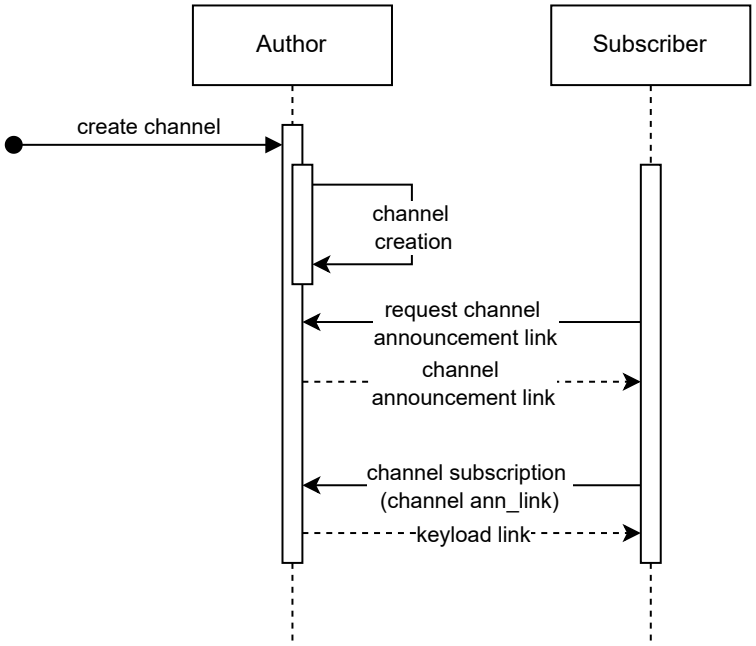


Figure 3.4: *IOTA Streams* channel creation and subscription process

request to the author, who can approve the subscription by replying with a *keyload message*. Only after this approval, the subscriber can read/write the channel using the keyload message.

By leveraging this approach, whenever an external user sends a request message to the IOTA Client, the system first checks whether it is authorized for one of the previously created channels. If so, the message, along with the user identifier, is stored in the corresponding channel. Upon successful completion of this process, the service request is forwarded to the Message Broker for execution on *L2*. If the external user is not authorized with respect to the corresponding IOTA Streams channel, then the request execution is denied. This approach leverages IOTA Streams channels as a method of user authentication, safeguarding the execution of services across the entire architecture.

3.4 Private semantic-enhanced blockchain layer

The dual-layer DLT-based architecture described in Section 3.2 can offer its architectural benefits while providing semantics-based services. A private blockchain layer realization on *L2* has been based on *SeeSaw* (SEmantic-Enhanced SAWtooth), [111] a semantic-enabled SOA for trustless collaboration in MEC and pervasive computing, particularly aimed at advanced CPS scenarios. It is based on the *Hyperledger Sawtooth* open source blockchain platform [85].

The SeeSaw proposal expands the distributed architecture of Sawtooth. Figure 3.5 shows the overall infrastructure of the private *L2* blockchain instance, whose key elements are outlined in what follows.

- ***Producer (P)***: publishing resources by registering them on the blockchain as assets.
- ***Consumer (C)***: requiring resources through a semantic-based process comprising Discovery, Composition, Negotiation, Explanation (optionally) and Selection.
- ***Web Interface (WI)***: collecting inputs from Producers and Consumers and forwarding to a Validator. A WI can manage multiple C and P instances; in the current implementation the *WebSocket* protocol [37] is adopted, which is suitable for resource-constrained IoT contexts [77], anyway further point-to-point or mesh protocols for IoT could be implemented. On the other side, the WI communicates with exactly one Validator through *ZeroMQ* (`http://zeromq.org/`) distributed messaging protocol.
- ***Transaction Processor (TP)***: executes transactions at the edge of the network implementing smart contracts. Sawtooth supports Transaction Processors written in Python, C++, Go, Java, JavaScript or Rust. In the proposed approach C++ has been chosen for the sake of efficiency. Each TP communicates with a Validator through ZeroMQ.
- ***Validator***: accesses the *radix Merkle tree* data structure of the blockchain. It receives transaction requests from a WI and dispatches

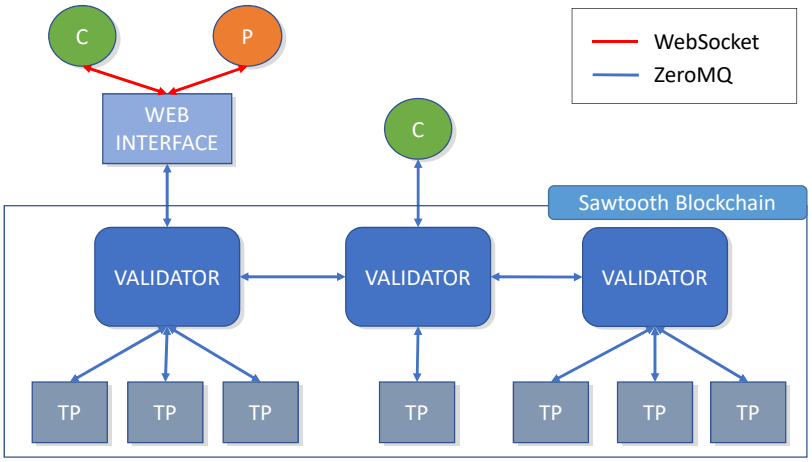


Figure 3.5: Framework architecture

them, balancing the load among all connected TPs, according to the manager/workers model. If a TP is fully loaded, a transaction is returned as invalid: WI typically implements a backoff mechanism for waiting before resubmission. Validators form a peer-to-peer network, communicating through a gossip protocol built on ZeroMQ for the replication of smart contract execution and the consensus protocol. Sawtooth adopts PoET: a Validator is elected as leader through a lottery-like approach and is allowed to add a new block of transactions to the chain. Sawtooth Validators are implemented in Python.

In this architecture, the WI component is responsible for aggregating requests coming from Producers and Consumers in a format suitable for processing by the Validator node, and then forward the resulting messages to it. Alternatively, as also illustrated in Figure 3.5, Consumers can directly forward properly pre-processed inputs to a Validator node as ZeroMQ messages.

Depending on target scenarios, several deployment configurations are possible. First of all, Producers and Consumers are components that require minimal computational requirements, thus they can be aimed at nodes deployed in the field (*e.g.*, mobile devices or embedded in cyber-physical system). WI nodes can be either dedicated devices at the edge of the network or integrated into Validators. Instead, Validators require the largest amounts of mass memory and bandwidth, mainly due to blockchain storage and consensus. They are the most suitable to be hosted on premises

at the core of the organization's network. Conversely, TPs do not need large storage or bandwidth, as they process one transaction at a time. They need relatively fast processing capabilities, but with proper smart contract optimization even low-cost single-board computers like *Raspberry Pi* may be a suitable platform. All exchanged messages are serialized in the *Protocol Buffers* format `https://protobuf.dev/`). Transactions can be processed and validated individually or in *batches*, depending on request parameters. A batch induces a sequential dependency relationship among transactions: if one fails, the subsequent ones are not processed and the whole batch is invalidated.

3.4.1 Knowledge representation and Smart Contracts

The proposed approach enables a semantic-based resource/service discovery and negotiation in an IoT-oriented blockchain. As a consequence, the blockchain itself can be considered as a SOA, implementing resource registration, discovery, and selection as SCs, with distributed execution and consensus-based validation.

The discovery, composition and negotiation services are based on semantic matchmaking of descriptions of requests and a set of resources, annotated as $\mathcal{ALN}$ DL concept expressions in OWL 2 w.r.t. a shared ontology. For each request-resource annotation pair, a *semantic relevance score* is computed from a combination of penalties induced by Concept Contraction and Concept Abduction [106] as recalled in Section 2.2.3. This introduces a formally founded relevance ranking of all available resources on the blockchain that are described w.r.t. the same ontology as the request. The adopted inference services also return a logical *explanation* of service outcomes, which improves understandability of results w.r.t. other types of approaches. Transactions are recorded on the blockchain for robustness, traceability and accountability purposes. SOA primitives and corresponding SCs are reported hereafter.

A. Registration. Several resource domains can co-exist in the same blockchain. Each domain is associated to a different ontology, which provides the reference conceptual vocabulary to annotate resources. Every ontology is identified by a unique URI, as per OWL specifications. Each node can own resource instances, characterized by:

- a URI identifying the resource unambiguously;
- a semantic annotation in OWL language, modeling high-level descriptive information of resource features;
- the URI of the reference ontology;
- a set of data-oriented attributes stored as a key-value pair, allowing to integrate and extend logic-based inferences with application-specific and context-aware information processing.

In order to make a resource available for discovery and usage, the owner registers it as an asset on the blockchain storage. Ontologies are stored in the same way. Through this SC a blockchain-backed u-KB is thus obtained.

B. Resource discovery. IoT-based applications vary widely in functional and Service Level Agreement (SLA) requirements. Some use cases need quick-response resource discovery and best-effort recall is tolerated; this is typical of pervasive computing contexts. Other applications need an exhaustive search space exploration to guarantee that the best possible resources are found. In order to cope with the widest range of scenarios, SeeSaw includes two discovery modes, named *fast* and *full*. Supposing n annotated resources are associated with a domain ontology, any discovery request will generate up to $k+1$ SC transactions, one for each piece originated from considering different sets of p resources from the whole set (*i.e.*, the ABox of the u-KB), with no overlap. Each of the first $k = \lfloor n/p \rfloor$ transactions will refer to exactly p resources and the last piece to the remaining $n - kp$ ones A transaction yields a *hit* if at least one of the resources in the piece has a semantic affinity higher than a given threshold, a *miss* otherwise. Hit resources are returned to the requester. In full discovery, all $k+1$ transactions are submitted simultaneously and the receiving Validator will take care of load balancing among Transaction Processors; the requester will receive all resources above the semantic relevance threshold. Furthermore, both hit and miss transactions are committed to the blockchain for traceability purposes. Conversely, fast discovery submits *clusters* of at most $c \leq k$ transactions at a time and it is limited by an overall *timeout*. As soon as a cluster returns a hit or when the timeout expires, remaining clusters are not submitted. Moreover, in fast discovery miss transactions

are invalidated and not committed to the chain, in order to reduce consensus stress on Validators. The adoption of clusters aims at a trade-off between a completely serial and parallel piece processing: the former minimizes blockchain load, but may increase the length and variability of hit latency, potentially incurring in more frequent timeouts; the latter ensures that a hit is found if it exists in the chain, but places a heavier computational burden and incurs in higher turnaround time.

Parameters of the discovery SC are:

- mode: `fast` or `full`;
- URI of the reference ontology: this determines the resource domain as well as the vocabulary used to express both the request and the resources to be retrieved;
- semantic annotation of the request in OWL language, specifying desired resource features and constraints;
- maximum acceptable value for the i^{th} data-oriented attribute $a_{i_{max}}$ (*e.g.*, the maximum price the requester is willing to pay). Resources with at least one value higher than this threshold will be skipped from matchmaking (thus reducing computational overhead);
- minimum semantic relevance threshold s_{min}, as a floating-point number in the $[0, 1]$ interval, with a value of 1 corresponding to a full match and 0 to a complete mismatch (both rare situations in realistic scenarios); after matchmaking, resources with a relevance score below this threshold will not be returned, as deemed irrelevant to the requester.

C. Explanation. This SC is used to request a motivation for matchmaking outcomes. This may be useful for request revision and refinement [106] as well as *post-hoc* audit of the discovery process. Explanation reinforces the overall trust in the blockchain framework not only at data management level, but also at application level. Parameters of the SC are:

- the request annotation;
- the URI of the discovered resource.

SC result consists of the semantic affinity score $0 \leq s_i \leq 1$ and concept expressions of G and K from Concept Contraction and of H from Concept Abduction.

D. Resource selection. After receiving all results exploiting full discovery, or a subset with fast discovery, the requester can select the best discovered resource with this SC. The complete registered resource representation is retrieved from the blockchain and returned. The proposal does not constrain resource fruition in any way, leaving application-specific details –such as interface endpoint or payment method– to resource annotations themselves.

E. Composition. Extension of the Discovery service based on the non-standard inference of *Concept Covering*. The Composition SC takes seeks to cover the request annotation as much as possible with all resource annotations available in the blockchain related to the same reference ontology.

F. Negotiation. In physical marketplaces, after a successful discovery but before resource fruition, provider and requester agents may need to engage in a negotiation. This could involve a buyer wishing to purchase from a seller, negotiating the characteristics of the product or service and the terms of the transaction until a satisfactory agreement is reached. Given negotiable descriptions of a request and a resource, an agreement is achieved if certain characteristics, endowed with specific utility, are relinquished, leading to a new compromise that deviates from the initial request but not so much as to make the purchase unacceptable for either party.

Using Semantic Web techniques and technologies, it is possible to model and execute automatic negotiation between two agents, referred to here as the *requester* and *provider*. This negotiation will revolve around semantic annotations respectively representing the request and an available resource (typically the one resulting from a preliminary Discovery request). When the requester wishes to make a purchase, *i.e.* access to a resource offered by the provider, they will send a request containing the semantically annotated description of the desired resource. The bargaining will be supported by the blockchain, applying a Smart Contract that internally performs all the steps of a bilateral negotiation algorithm to reach an agreement. The bilateral nature of the negotiation means that the requirements and preferences of both the requester and the provider are deemed equally important [100].

However, it is not guaranteed that an acceptable agreement for both parties will always be reached. For this reason, a semantic approach capable of providing a precise and rigorous explanation of the negotiation process is useful: the system will display the components each user must give up relative to their initial proposal. When the negotiation process ends with an agreement, it will be beneficial (positive utility) for both agents. Concept Contraction is used to calculate the conflicting information during each negotiation round.

The negotiation process is carried out by assigning a weight, known as *utility*, to each element of the semantic annotation of both the resource and the request. The sum of all weights of a single annotation must be 1. For each agent at the beginning of the negotiation, a disagreement threshold is also defined, which will regulate whether the counterpart's proposal is acceptable or not and will dictate when the agent decides to exit the negotiation with a failure. In detail, the SC follows a multi-attribute bilateral negotiation protocol for mobile agents, consisting of the following key phases:

1. Concept Contraction is executed alternately, reversing the annotations of the requester and the provider. This allows to compute the Contraction of the provider's annotation relative to the requester's one and vice versa. Based on the principle of asymmetric match in semantic matchmaking, these are not symmetrical operations and will return different *Give Up* and *Keep* elements, implying the recalculation of the total utility for both agents and the reformulation. The two *Give Up* concepts calculated represent the conflicting elements between the buyer and vendor, which can be negotiated to seek an agreement;

2. The total utility is computed to determine which element should be removed at each round to attempt to complete the negotiation. This operation is performed for both agents;

3. It is verified that the total utility is greater than the disagreement threshold for both agents;

4. For both agents, it is checked whether the *Give Up* contains elements. If so, the annotation is reformulated to attempt to conclude the ne-

gotiation with an agreement by removing the elements indicated in the *Give Up* concept expression, *i.e.* the element with the currently lowest utility, and total utility is reduced by the utility value of the removed element;

5. Steps 1-4 are repeated until one of the following conditions is met:

   - Agreement: There are no more conflicting elements to negotiate (both *Give Up*s are empty), indicating an agreement between the parties and a successful negotiation outcome;
   - Disagreement: for one of the parties, the total utility is less than the disagreement threshold, leading to the conclusion of the negotiation with a disagreement (this situation can occur when an element with high utility must be relinquished, making the lack of it unacceptable).

This structured approach ensures that the negotiation process is systematic and aimed at maximizing the possibility of reaching a mutually acceptable agreement while minimizing the information each party must relinquish. The implementation by means of a SC stored in the blockchain guarantees both parties against cheating and on the correctness of the final outcome, whatever it may be.

3.4.2 Integration into the two-layer DLT architecture

The described semantics-enhanced blockchain is easily integrated into the two-layer DLT architecture. A network of blockchain platforms that implement the SCs described in Section 3.4 must be registered on the two-layer DLT infrastructure, enabling distributed computation. According to the architecture described in Section 3.4, the *Web Interface* serves as the Manager of the *Seesaw* network in the DLT infrastructure. Thus, the Manager is able to register with the Gateway by sending all details involving the provided semantic-based services. *Producers* and *Consumers* can interact through the *IOTA Client* component of the system. As a result, multiple semantic-based blockchain infrastructures can be deployed on *L2* to handle all services described in Section 3.4.1 in a federated fashion. This can further increase horizontal scalability in addition to the request partitioning algorithm provided by SeeSaw.

Federation of several SeeSaw instances can be setup in various ways. In one possible approach, each one of the SeeSaw instances in *L2* can be deployed to manage a specific subset of resources for a given wider domain. In this scenario, when a user submits a request to the IOTA Client, it will trigger the SC execution on multiple semantics-enhanced blockchains in *L2*. Subsequently, the separate results from these different executions will be aggregated to produce a comprehensive response to the user. On the other hand, federation can be leveraged to handle multiple different domains within a single system. This creates a versatile, multi-purpose solution, while still maintaining clear separation of concerns. Knowledge describing different domains is handled by different platforms by means of distinct ontologies, thereby preserving domain specificity while ensuring system scalability and adaptability.

The following section details the complete interaction flow for the semantic resource Discovery service within the two-layer DLT architecture, serving as a clarifying example of how semantic SCs are integrated into the proposed two-layer architecture.

1. The **user** sends a request to the **IOTA Client** contacting the appropriate endpoint of the ReST API. Internally, the ReST API performs the following steps:

    - insertion of the request into the dedicated *IOTA Streams* channel. The user who is inserting the message into the private channel uses the *keyload message* obtained when the subscription process was completed;
    - if the insertion of the request message into the channel succeeds, then it it published on the respective topic of the **Message Broker**. Before performing this step, it computes a **correlationID** inserted into the message and stored locally, together with the details on how to contact the user when a response is ready;
    - if it is the first time that a request is posted on a given topic, the associated reply topic is created and the IOTA Client issues a subscription;

2. the **Gateway** is notified that the request is available on one of the topics it (created and) subscribed. It retrieves the message and it ex-

tracts the name of the topic. Based on the name of the topic, it sends it to the appropriate **Manager** that is in charge of the *L2* blockchain network. In order to reach the correct Manager, it leverages the *topic name–networkURL* internal mapping and it passes the request to the appropriate endpoint for request reception on the Manager;

3. the Manager that received the request message invokes the SC instances on the connected blockchain(s), using libraries for the specific platforms it interfaces with.

   When a user issues a *service discovery* semantic request, the URI of the reference ontology is specified in the request message. Using this information, the Manager component maintains a mapping of all connected blockchains that manage resources related to that specific ontology, thereby identifying the blockchain(s) responsible for the specific domain or sub-domain. The Manager is able to contact the correct SC instances to execute the *discovery* process across all available resources. Each SC executes the algorithm described in 3.4, resulting in a *hit* if one of the resources achieves a semantic score higher than a given threshold or a *miss* otherwise.

   The key distinction between integrating this algorithm within a single blockchain platform and leveraging the two-layer DLT infrastructure lies in the physical distribution of domain resources across different platforms, rather than forcing a logical separation on a single blockchain when the *fast* mode is required. This enables the implementation of a *u-KB* approach which offers the following benefits:

   - higher scalability: resources are not stored together on a single platform, but they are scattered on different blockchains, reducing the burden in terms of storage on any single blockchain that implements semantic services. This approach allows for the addition of new resources without concerns for the growing size of an ontology stored on a single blockchain;
   - lower computational load: the overall computation associated with resource handling during the execution of the semantic algorithm can be simplified. The need for subdivision into pieces can be reduced or deactivated in the SCs, since this process is

handled physically and overseen directly by the Manager implementation.

4. the **Gateway**, which previously invoked the manager, receives the result of the invoked service. In the case of a user requesting the Discovery service, this response includes the set of resources retrieved, if there was a hit on the resources available on the blockchain(s) that executed the SC. The result is inserted in the response topic and it is encapsulated in a message containing the *correlationID* computed in the first step;

5. at level *L1*, the IOTA Client, listening on the topic on which the reply is inserted, takes the incoming message with the resources obtained from the semantic SC and performs the following steps:

   - tracks the response on the same IOTA Streams channel where the request was placed;
   - leveraging the *correlationID*, it obtains the connection details to reach the user and sends back the response.

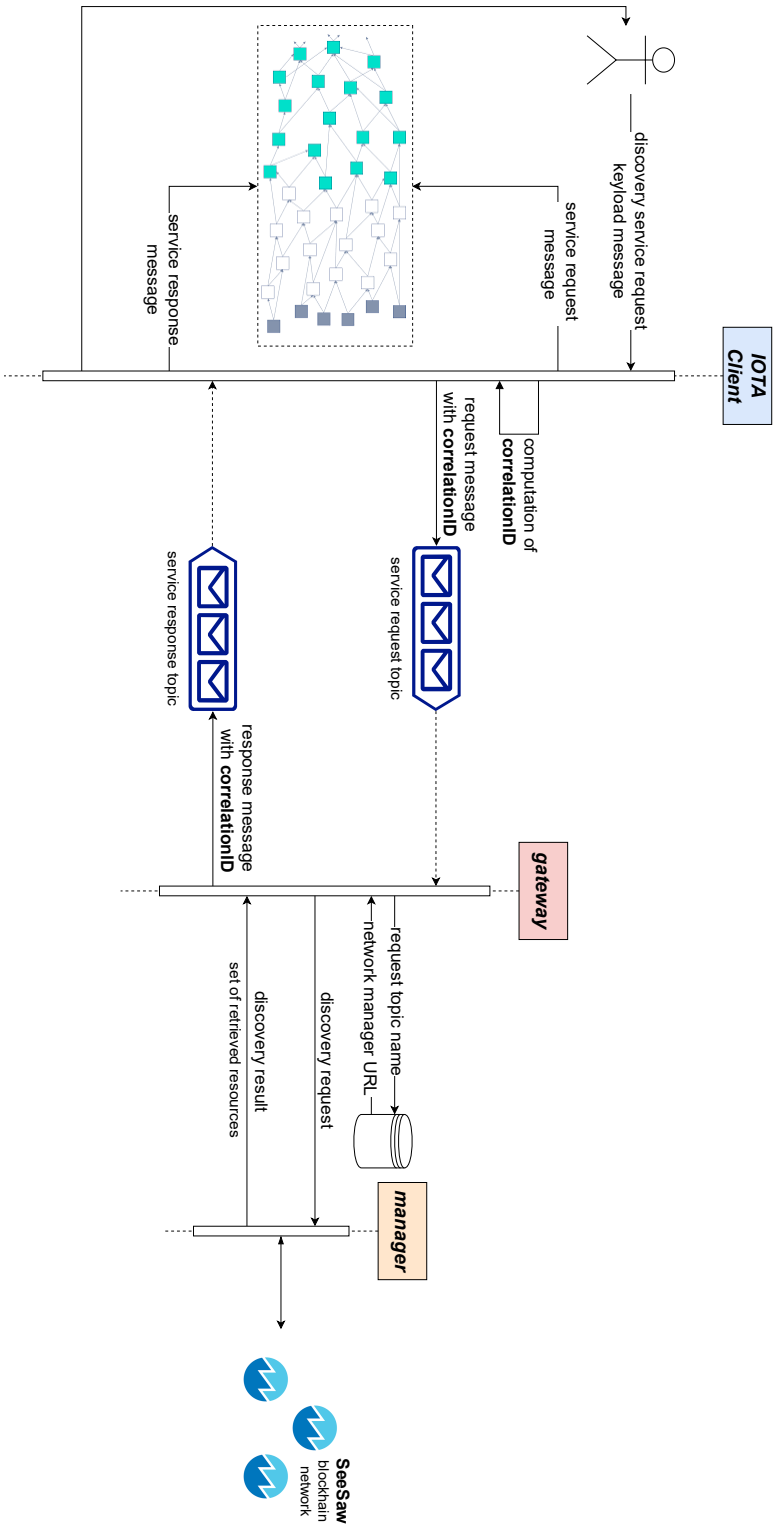


Figure 3.6: Interactions in the semantic-based dual-layer DLT

Chapter 4

Argumentative decision-making framework for Multi-Agent Cyber-Physical Systems

The rapid expansion of the IoT has increased the number of agents embedded within environments capable of recognizing situations, interpreting events, coordinating actions, and collaboratively solving problems. This requires flexible distributed cooperation in complex CPSs including heterogeneous components. Within this perspective, the SWoE envisions a new generation of intelligent agents operating on personal devices, designed to express, aggregate, and process meaningful information fragments by means of interoperable coordination with local or remote hosts [13]. As explained in Chapter 2, paradigms such as the Social IoT and the SWoE can play a pivotal role.

In a CPS, sensors and actuators collect data from the physical world and execute control decisions based on that data. Therefore, coordination in a CPS requires that agents not only annotate sensed data with rich, formal semantics, but also exchange information to manage agreements and disagreements concerning perceptions as well as alternative decisions. While Chapter 3 focuses on advanced communication and coordination among IoT devices leveraging a novel DLT architecture to provide the semantic-based services of discovery, composition and negotiation, this chapter describes

an AI-based MAS layer to be deployed on top of it, in order to allow agents to interact autonomously and make meaningful and informed decisions in real-time, minimizing human intervention and establishing feedback loops where computations influence physical processes and, conversely, physical processes affect computational outcomes.

Specifically, the challenge of modeling and executing multi-agent discussions is addressed by the *Computational Argumentation* framework. Specifically, Dung's Argumentation Framework (AF) [32] provides a graph-based formalism for reasoning over conflicting knowledge by representing arguments as atomic information units, disregarding their internal structure, and focusing solely on their attack relations, which denote conflicts between arguments.

In a CPS scenario, as new information emerges, agents represent their asserted knowledge as defeasible arguments whose validity may be contested by others. To facilitate this, arguments are aggregated and organized into a graph structure where edges denote the relationships between argument pairs. Building on this theoretical foundation, two primary families of approaches for argument evaluation have been proposed in the literature

- Abstract Argumentation (AA) [32] evaluates the acceptability of each argument solely based on their interrelations, abstracting away from the content of the arguments themselves;
- Structured Argumentation (SA) [15] adopts formal models to represent arguments and applies inferences to assess relations.

This chapter introduces a general-purpose framework for SA that harness KRR to represent arguments, evaluate relationships, and determine their acceptability by exploiting non-standard inference services over information fragments expressed in DLs, which support approximate matches and explanation of results. Specifically, the framework aims to define an ontology-based argumentation framework where OWL annotations are used not only for modeling information, but also to introduce a process, based on semantic matchmaking, identifying and weighting relations through non-standard inferences. The framework components are designed for computational efficiency, facilitating implementation across diverse CPS contexts.

4.1 Motivation

The application of Dung's argumentation theory has notably expanded in recent years within CPS scenarios, primarily to model interactions among intelligent agents and to enable self-coordination grounded in argumentation. For instance, the study presented in [68], employs argumentation to orchestrate smart vehicles navigating congested roadways. In this setting, arguments encapsulate both the data harvested from vehicle sensors and the array of possible actions available to each vehicle. By analyzing the argumentation graph, each vehicle agent can resolve conflicts, discern predominant arguments, essentially the recommended actions, and adjust lane positions in alignment with current road conditions. Similarly, the work in [64] utilizes argumentation graphs to model object interactions akin to natural language dialogues. Through case studies focusing on traffic management and ambient-assisted living, this research aims to develop an argumentation-based decision-making system that addresses the limitations inherent in traditional rule-based approaches within IoT environments. While both studies effectively demonstrate the practicality and benefits of incorporating argumentation in IoT, they do not provide formal argument models or methodologies for evaluating relationships among arguments. Addressing conflict resolution from a different perspective, [39] proposes a game-theoretic weighted voting scheme within the context of Social IoT. In this approach, each smart object votes for or against specific arguments to reach an optimal conclusion. However, this method is constrained by its inability to allow each social object to express a nuanced, graduated vote for each identified conclusion, instead relying on a basic binary "yes/no" voting mechanism.

The integration of KRR methods, such as those provided by Semantic Web technologies, enhance the potential for developing effective argumentation frameworks to manage a multitude of agents. However, this integration also presents challenges related to formal property characterization, explainability, computational feasibility, and practical applicability. In this context, early research leverages AA to reason over inconsistent KBs. Notably, [73] introduces the *Generalized Argumentation Framework* (GenAF) , which constructs an argumentation graph upon an underlying KB. In GenAF, each argument atom represents a formula within the

KB, and attack relations between arguments model inconsistencies stemming from conflicting information sources. Although GenAF provides a versatile extension to abstract argumentation, adaptable to various logics for knowledge representation, it relies exclusively on consistency checks, specifically satisfiability assessments of argument conjunctions, for conflict recognition. Further advancing this line of work, [20] proposes a deductive argumentation framework aimed at reasoning with conflicting and uncertain ontologies. This framework incorporates two distinct relations within its argument structures and, unlike GenAF, assigns a weight to each argument to indicate the degree of information certainty. Despite these contributions, both approaches utilize the final argumentation graph to evaluate the acceptability of arguments without offering implementations in concrete scenarios. Early proposals like DILIGENT [129] acknowledge the potential of OWL-based formal models of argumentation, but express skepticism regarding their applicability among groups of human agents. Furthermore, previous works do not integrate the underlying logical formalisms with standard Semantic Web languages, resulting in the absence of a comprehensive and practical framework.

This dissertation introduces a framework where agents generate and share semantic annotations using standard Semantic Web languages grounded in DLs [8], specifically referencing domain ontologies. The framework adopts a subset of OWL 2 [89] that corresponds to the $\mathcal{ALN}$ DL, which supports polynomial-time standard and non-standard inferences on acyclic, "bushy but not deep" ontologies [80]. Aligned with Dung-style AA [32], each argument is represented through semantic-based metadata. It employs a Bipolar Weighted Argumentation Framework (BWAF) [3], which enables a nuanced characterization of argument relations by assigning weights that indicate both their type (attack or support), and their strength. The appraisal of relations leverages non-standard, non-monotonic logic-based inferences, namely *Concept Contraction*, *Concept Abduction* and *Concept Bonus*, each equipped with formal explanations of their outcomes [105]. After evaluating relations within the argumentative graph, the framework computes an acceptability score for each argument using a novel propagation-based gradual semantics. This approach produces more refined assessments of arguments by assigning numerical scores, allowing for arguments to be ranked from the most to the least acceptable [2]. This contrasts with tradi-

tional applications that label arguments with a simplistic *accepted/rejected* evaluation, which is inadequate for accurate decision-making by autonomous agents. To address potential computational limitations of agents, the proposed method incorporates a *fading* mechanism [18] into path strength propagation as well. This mechanism accounts for the diminishing effectiveness of long argumentation paths, thereby reducing the storage and computational demands on devices in CPS environments. Moreover, unlike many existing approaches, the proposed framework effectively handles cyclic argumentative graphs by employing an iterative algorithm with defined halt conditions. This ensures robustness and reliability in the argumentation process, even in complex graph structures.

4.2 Bipolar Weighted Argumentation Framework

A BWAF [3] integrates two significant extensions of Dung's AFs: the bipolar AF (BAF) and the weighted AF (WAF).

First of all, in a BAF [21], arguments can interact in two distinct ways: through attack or support relations. Additionally, new forms of attacks arise from the chaining of direct attacks and support relations. For instance, a *supported attack* is characterized by a chain of supports culminating in an attack, whereas an *indirect attack* involves an attack followed by a series of supports. Instead, in WAFs [33], weights are associated with relations to indicate their relative strength. Thus, the BWAF framework merge the two theories by defining attacks and support relations and incorporating weights for both. Specifically, each support is assigned a positive weight, and each attack receives a negative one, allowing for both weighted attacks and weighted supports.

The formal definition can be stated as follows:

Definition 1 (Bipolar Weighted Argumentation Framework) *A BWAF is a triple* $\mathcal{G} = \langle \mathcal{A}, \hat{\mathcal{R}}, w_{\hat{\mathcal{R}}} \rangle$*, where* $\mathcal{A}$ *is a finite set of arguments,* $\hat{\mathcal{R}} \subseteq \mathcal{A} \times \mathcal{A}$ *and* $w_{\hat{\mathcal{R}}} \colon \hat{\mathcal{R}} \mapsto [-1, 0[\cup]0, 1]$*. Attack relations are defined as* $\hat{\mathcal{R}}_{att} = \{\langle a, b \rangle \in \hat{\mathcal{R}} \mid w_{\hat{\mathcal{R}}}(\langle a, b \rangle) \in [-1, 0[\}$ *and support relations as* $\hat{\mathcal{R}}_{sup} = \{\langle a, b \rangle \in \hat{\mathcal{R}} \mid w_{\hat{\mathcal{R}}}(\langle a, b \rangle) \in]0, 1]\}$*. Given two arguments* $a, b \in \mathcal{A}$ *and a path* $\langle a, x_1, x_2, \ldots, x_n, b \rangle$ *from* a *to* b*, then:*

- a bw-attacks b *if* $w_{\hat{\mathcal{R}}}(\langle a, x_1\rangle) \cdot w_{\hat{\mathcal{R}}}(\langle x_1, x_2\rangle) \cdot \ldots \cdot w_{\hat{\mathcal{R}}}(\langle x_n, b\rangle) < 0$.
- a bw-supports b *if* $w_{\hat{\mathcal{R}}}(\langle a, x_1\rangle) \cdot w_{\hat{\mathcal{R}}}(\langle x_1, x_2\rangle) \cdot \ldots \cdot w_{\hat{\mathcal{R}}}(\langle x_n, b\rangle) > 0$.

In the context of a directed weighted graph, arguments are represented as semantic annotations that agents exchange, expressed as concept expressions in $\mathcal{ALN}$ (unfolded and normalized into *Conjunctive Normal Form*) relative to a scenario-dependent ontology $\mathcal{T}$. The collection of annotations exchanged by agents corresponds to the set $\mathcal{A}$ in the BWAF $\mathcal{G}$, while the pairwise interactions between agents align with the set of relations $\hat{\mathcal{R}}$ between the respective arguments. More formally, consider two generic agents A_R and A_S in a network (*e.g.*, smart devices in a pervasive computing environment). The annotations associated with these agents – denoted as R and S respectively – are interpreted as arguments, since they represent the conclusions reached after internal information processing. If A_S communicates with agent A_R ($A_S \rightsquigarrow A_R$), the proposed matchmaking-based approach considers R as request and S as resource. This is inspired by CPS MASs such as [113], where S can be employed to "respond to needs" expressed by R. This feature illustrates how an argumentative relationship might be established, with the directional edge pointing from S to R.

Once the direction of a relation is established, the algorithm sketched in Figure 2.2 assesses its type, determining whether S performs an attack or a support towards R. For this purpose, a semantic *consistency check* is conducted as the first step: if $R \sqcap S$ is satisfiable w.r.t. the ontology $\mathcal{T}$, the relation from S to R is classified as a support; otherwise it is categorized as an attack.

$$w_{\hat{\mathcal{R}}}(\langle S, R\rangle) = \begin{cases} w_{\hat{\mathcal{R}},att}(\langle S, R\rangle) & if\ \mathcal{T} \models R \sqcap S \sqsubseteq \bot \\ w_{\hat{\mathcal{R}},sup}(\langle S, R\rangle) & otherwise \end{cases} \tag{4.1}$$

For each case, specific strategies are outlined in Equation 4.1 to assign appropriate weights to the $S \rightsquigarrow R$ edge, leveraging non-standard inference services. This process is repeated for each pair of nodes that interact within the graph.

4.2.1 Attack relation weight assessment

When an inconsistency between the semantic annotations of two arguments is detected, an attack relation arises only. The next step involves computing

its weight taken into account the following informative contributions (here, the attacker argument corresponds to the resource S and the attacked argument is the request R):

1. the amount of conflicting information between the two arguments;
2. the amount of information that is confirmed by both arguments;
3. the amount of the information in the attacked argument that is neither confirmed nor refuted by the attacker;
4. any additional information present in the attacker that is absent in the attacked argument.

This information is derived through the application of *Concept Abduction*, *Contraction* and *Bonus* (Section 2.2.3). The final formula for the attack's weight must account for the scores generated by all these non-standard inference services. Moreover, each of the contributions mentioned above is expressed as a signed term in an algebraic sum. In general, the weight of an attack in BWAF, must be a real negative value. Therefore, the term representing the conflicting information (term 1) is assigned the greatest importance and carries a negative sign. In contrast, the remaining three contributions serve to intuitively reduce the attack's overall strength, and thus are assigned positive signs. This leads to the following formula:

$$w_{\hat{\mathcal{R}},att}(\langle S, R\rangle) = -\alpha\frac{p_c(R,S)}{\|R\|} + \beta\frac{\|Bonus(Bonus(K,S),S)\|}{\|R\|} + \gamma\frac{p_a(K,S)}{\|R\|} + \delta\frac{\|Bonus(R,S)\|}{\|R\|\cdot\|S\|} \tag{4.2}$$

where p_c and p_a are the penalties of *Concept Contraction* and *Abduction* (see Section 2.2.3), and $\|\cdot\|$ is the CNF norm. Coefficients allow giving different emphasis to each term, as explained above.

Given an inconsistency between R and S, the amount of conflicting information is given by *Concept Contraction* penalty $p_c(R,S)$. In detail, it is computed as norm of the *Give Up* concept: $p_c(R,S) = \|G\|$. That term is normalized by $\|R\|$, *i.e.*, the worst-case value of $p_c(R,S)$ (all the information in R is rebutted by S).

The second term in Equation 4.2 takes into consideration:

- the additional information in S w.r.t. K *i.e.*, the consistent part of R obtained from *Concept Contraction*;
- what is "not additional", *i.e.*, common to both K and S.

This is computed via a nested pair of Bonuses, with the inner $Bonus(K, S)$ gives the first contribution, and the outer Bonus corresponds to the second.

The third term considers the part of R which is neither in conflict nor confirmed w.r.t. S: this comes from the penalty p_a of Concept Abduction between *Keep* K and S: it is computed as $\|H\|$, where the *Hypothesis* H is the part of K not matched by S.

Both the second and third terms are normalized by $\|R\|$, which is the maximum possible value.

Finally, the last component in Equation 4.2 quantifies the additional information the attacking argument S has w.r.t. to the attacked one R; in this case the normalization is by the product of the norms of R and S.

4.2.2 Support relation weight assessment

As previously discussed, an argument S can only support R if there is no conflict between them. In typical matchmaking scenarios, where R represents the request and S the resource, the support relation is determined by two factors:

1. the amount of information lacking in S to reach a full match with R;
2. the amount of additional information S has w.r.t. R.

This leads to the following formula:

$$w_{\hat{\mathcal{R}},sup}\left(\langle S, R\rangle\right) = 1 - \frac{p_a(R,S)}{\|R\|}\left(1 - \frac{\|Bonus(R,S)\|}{\|S\|}\right) \tag{4.3}$$

where p_a and Bonus have been already defined. The maximum support of S to R occurs when $\mathcal{T} \models S \sqsubseteq R \Leftrightarrow H \equiv \top \Rightarrow p_a(R,S) = 0$, whereas support is minimum when $H = R \Rightarrow p_a(R,S)/\|R\| = 1$. The last term of Equation 4.3 quantifies the additional information of S w.r.t. R through the Bonus inference, analogously to the Attack formula.

4.3 Propagation-based ranking semantics

4.3.1 Definitions

In this paragraph, a set of foundational definitions in BWAF are introduced.

The concept of *path length* is important to quantify the directness or indirectness of the influence from one argument to another, thereby affecting the propagation of argumentative strength across the network.

Definition 2 (Path length) *Let $\mathcal{G} = \langle \mathcal{A}, \hat{\mathcal{R}}, w_{\hat{\mathcal{R}}} \rangle$ be a BWAF and $x_1, x_n \in \mathcal{A}$ two arguments s.t. there exists a path $p^* = \langle x_1, x_2, \ldots, x_n \rangle$. The length i of the path p^* is the number $n-1$ of relations $\langle x_{j-1}, x_j \rangle$, where $j = 2, \ldots, n$.*

The path length in BWAF defines the number of relations between two arguments. Since every single relational step is associated with a weight, *strength propagation* aggregates the strengths of weighted relations along a path to determine the overall influence from the initial argument to the terminal one.

Definition 3 (Strength Propagation) *Let $\mathcal{G} = \langle \mathcal{A}, \hat{\mathcal{R}}, w_{\hat{\mathcal{R}}} \rangle$ be a BWAF and $x_1, x_n \in \mathcal{A}$ two arguments such that there exists a path $p^* = \langle x_1, x_2, \ldots, x_n \rangle$. The* strength propagation *(sp) from x_1 to x_n for p^* is given by:*

$$sp\left(x_1, x_n\right)_{p^*} = \prod_{i=2}^{n} w_{\hat{\mathcal{R}}}\left(\langle x_{i-1}, x_i \rangle\right) \tag{4.4}$$

This definition computes the strength of the path p^* by multiplying the weights of all constituent relations (regardless of their direction) between successive arguments along the path. By construction, the value of $sp\left(x_1, x_n\right)_{p^*}$ lies within the interval $[-1, 0[\cup]0, 1]$, serving as a quantitative measure of the influence that x_1 exerts on x_n. A positive value of $sp\left(x_1, x_n\right)_{p^*}$ indicates a support (*bw-supports*) relation from x_1 to x_n, while a negative value signifies an attack (*bw-attacks*) relation.

Furthermore, the significance of different paths converging on a particular argument x_n is also determined by the nature of the initial argument x_1 of each path. This observation leads to the following definition.

Definition 4 (Connected and free arguments) *Let $\mathcal{G} = \langle \mathcal{A}, \hat{\mathcal{R}}, w_{\hat{\mathcal{R}}} \rangle$ be a BWAF. The set of arguments $\mathcal{A}$ is partitioned in two disjoint subsets:*

- $\mathcal{A}^c = \{y \mid y \in \mathcal{A} \wedge \exists x \in \mathcal{A} \text{ s.t. } \langle x, y\rangle \in \hat{\mathcal{R}}\}$ *is the subset of arguments which receive at least one attack or support, denoted as* connected *arguments;*
- $\mathcal{A}^f = \mathcal{A} \backslash \mathcal{A}^c$ *is the subset of arguments in* $\mathcal{A}$ *which are not attacked or supported by any other argument, hereinafter called* free *arguments.*

Moreover, the relative significance of different paths that culminate at a particular argument is also influenced by the classification of the originating node, as detailed in Definition 4.

Definition 5 (Free-born and connected-born paths) *Let* $\mathcal{G} = \langle \mathcal{A}, \hat{\mathcal{R}}, w_{\hat{\mathcal{R}}} \rangle$ *be a BWAF and* $\alpha, \theta \in \mathcal{A}$ *two arguments. Any path* $p^* = \langle \alpha, \ldots, \theta \rangle$ *of any length* i *ending in* θ *belongs to one of the following two disjoint sets:*

- *Free-born paths* $\mathcal{P}_i^f(\theta)$*, starting in an argument* $\alpha \in \mathcal{A}^f$
- *Connected-born paths* $\mathcal{P}_i^c(\theta)$*, starting in an argument* $\alpha \in \mathcal{A}^c$

By analyzing a specific argument and all possible paths of any length that converge upon it, we can further classify each of these paths based on both their type and their associated strength propagation.

Definition 6 *Let* $\mathcal{G} = \langle \mathcal{A}, \hat{\mathcal{R}}, w_{\hat{\mathcal{R}}} \rangle$ *be a BWAF,* $\theta \in \mathcal{A}$*,* $\mathcal{P}_i^f(\theta)$ *and* $\mathcal{P}_i^c(\theta)$ *the sets of Free-born paths and Connected-born paths of any length* i *ending in* θ*. Each path* $p^* \in \mathcal{P}_i^f(\theta)$ *can be assigned to:*

- *Support Free-born paths set* $\mathcal{P}_i^{f+}(\theta)$*, if* $sp\,(\cdot)_{p^*} \in]0, 1]$*, or*
- *Attack Free-born paths set* $\mathcal{P}_i^{f-}(\theta)$*, if* $sp\,(\cdot)_{p^*} \in [-1, 0[$

Similarly, each path $q^* \in \mathcal{P}_i^c(\theta)$ *is in:*

- *Support Connected-born paths set* $\mathcal{P}_i^{c+}(\theta)$*, if* $sp\,(\cdot)_{q^*} \in]0, 1]$*, or*
- *Attack Connected-born paths set* $\mathcal{P}_i^{c-}(\theta)$ *if,* $sp\,(\cdot)_{q^*} \in [-1, 0[$

Consequently $\mathcal{P}_i^f(\theta) = \mathcal{P}_i^{f+}(\theta) \cup \mathcal{P}_i^{f-}(\theta)$ *and* $\mathcal{P}_i^c(\theta) = \mathcal{P}_i^{c+}(\theta) \cup \mathcal{P}_i^{c-}(\theta)$*.*

4.3.2 Algorithm

In the proposed BWAF ranking semantics, the assessment of an argument's acceptability is performed iteratively, as outlined in the Algorithm in Figure 4.1. The proposed ranking semantics takes into consideration the *fading* principle [18]. This principle states that that the influence of a chain of arguments on the final argument should decrease inversely with the chain's *length*. This stems from the observation that, in typical dialogues, longer chains of reasoning tend to be less persuasive, and this diminished effectiveness is due to the limitations of short-term memory. Since most pervasive computing devices operate under strict memory and storage constraints, applying the *fading* principle allows computational agents to effectively "forget" arguments that are temporally or spatially distant, ensuring that more immediate and relevant arguments have a greater impact on the acceptability assessment.

Formally, the algorithm outlined in Figure 4.1 takes into consideration a BWAF $\mathcal{G} = \langle \mathcal{A}, \hat{\mathcal{R}}, w_{\hat{\mathcal{R}}} \rangle$, and the constants $\zeta \in]0, 1[$, $\epsilon \in \mathbb{R}^+$. Moreover, it requires preliminary computation of $\mathcal{A}^f$, $\mathcal{A}^c$. The final result consists in the ranking of arguments $R = \{\langle a_i, s_i \rangle\}$, where $a_i \in \mathcal{A}$ and $s_i \in [-1, 1]$. Each argument $a_i \in \mathcal{A}$ is associated with an acceptability score, a real number in the $[-1, 1]$ interval, where 0 stands for neutrality, and higher (respectively, lower) values proportionally label acceptable (resp. unacceptable) arguments.

The iterative process begins once the arguments in the graph have been annotated with DL descriptions, and the relational edges have been both directed and assigned weights. Is is initially assumed neutral acceptability for all arguments: $\forall a_i \in \mathcal{A} : s_0(a_i) = 0$. This is also the final score for free arguments.

In the subsequent loop, at each iteration step $j > 0$, the procedure takes into account paths of length j. For each path length, the algorithm iterates over arguments a_i in the set of connected arguments $\mathcal{A}^c$. First of all, the *retrievePaths* function collects all free-born and connected born paths (defined as per Definition 5) with length j ending in a_i, $p^* = \langle x_1, x_2, \ldots, x_j, x_{j+1} = a_i \rangle$, suitable to update the acceptability score of said argument according to these two constraints:

(C1) the source argument x_1 and any other argument x_k, with $k = 2, 3, \ldots, j$

are different from the argument $x_{j+1} = a_i$ whose acceptability is being evaluated;

(C2) for each argument b repeated $n \geq 3$ times in the path p^*, all the $n-1$ sequences of arguments included between pairs of occurrences of b are different from each other.

These restrictions are imposed to ensure termination and make the algorithm less computationally expensive. Specifically, Constraint $(C1)$ is imposed because (i) the *sp* of any path originating from an argument θ must not impact the computation of the acceptability rank of the same argument θ, and, (ii) if the same θ is in a position $k < j+1$ of path p^*, it is plausible that the contribution of the path to the rank of θ has already been considered in one of the previous iterations. Constraint $(C2)$ prevents infinite loops, instead, by discarding paths with repeated sub-paths.

Once paths are collected, the following value is computed:

$$s_j(a_i) = s_{j-1}(a_i) + X + Y \tag{4.5}$$

where:

$$X = \begin{cases} \frac{\zeta^j}{|\mathcal{P}_j^f(a_i)|}(|\mathcal{P}_j^{f+}(a_i)| \sum_{p \in \mathcal{P}_j^{f+}(a_i)} sp(\alpha, a_i) \\ \quad +|\mathcal{P}_j^{f-}(a_i)| \sum_{p \in \mathcal{P}_j^{f-}(a_i)} sp(\alpha, a_i)) & if\ \mathcal{P}_j^f(a_i) \neq \varnothing \\ 0 & otherwise \end{cases}$$

$$Y = \begin{cases} \frac{(1-\zeta)^j}{|\mathcal{P}_j^c(a_i)|}(|\mathcal{P}_j^{c+}(a_i)| \sum_{p \in \mathcal{P}_j^{c+}(a_i)} sp(\alpha, a_i) \\ \quad +|\mathcal{P}_j^{c-}(a_i)| \sum_{p \in \mathcal{P}_j^{c-}(a_i)} sp(\alpha, a_i) & if\ \mathcal{P}_j^c(a_i) \neq \varnothing \\ 0 & otherwise \end{cases}$$

with $sp(\alpha, a_i)$ as per Equation 4.4 and $0 < \zeta < 1$ a coefficient having two purposes: satisfy the fading principle (by means of the exponential) and give greater weight to free-born paths. The latter idea is justified by the *void precedence* property in ranking semantics [2]. In addition to path admissibility, in order to prevent non-termination, the acceptability score is further checked. In fact, if the absolute value of the difference between $s_{i+j}(a_i)$ and $s_j(a_i)$ is less than a constant ϵ for two consecutive iterations,

score convergence is accepted and the procedure stops.

At the end of the algorithm, the *convertRanking* function is executed to ensure the convergence of the acceptability scores of each argument of a particular BWAF graph in the interval [-1,1] and to extract quantitative information on the acceptability of the arguments from the ranking. Let f be the logistic function and $x = s_i$ the score of each argument a_i computed by the algorithm; then $convertRanking(x) = 2f(kx) - 1$, with k sigmoid smoothing factor.

Excluding GP9, the proposed ranking semantics aligns with all the *group properties* (GPs) identified in [9]. These GPs group together conceptually related properties from the literature based on common patterns observed in gradual argumentation frameworks. The deliberate non-compliance with GP9 stems from the design choice of assigning an equal base score to all arguments within the bipolar weighted graph.

A crucial aspect of this approach is the variable number of iterations required to compute the acceptability scores of arguments in a BWAF. Free arguments, those without incoming edges, are immediately assigned a score of zero, eliminating the need for further computation. In contrast, connected arguments necessitate a number of iterations that depends on factors such as the number and length of paths leading to them, as well as specific parameters associated with these paths. By employing this methodology, the proposed semantics induces a gradual ranking of arguments based on their acceptability within a BWAF. This relationship is formalized in the following property:

Property 1 *For any BWAF $\mathcal{G}$, the propagation-based semantics $\mathcal{S}(\cdot)$ assigns a ranking relation $\succeq^{\mathcal{S}}_{\mathcal{G}}$ over the set of arguments $\mathcal{A}$, such that $\forall\ a, b \in \mathcal{A}$, $a \succeq^{\mathcal{S}}_{\mathcal{G}} b$ iff $\mathcal{S}(a) \geq \mathcal{S}(b)$.*

Consequently, the ranking procedure ensures a nuanced evaluation of argument acceptability within a BWAF. The principal features of adopting this semantics can be summarized as follows:

- capability to handle complex graphs: the approach can process bipolar weighted graphs that include cycles and paths of arbitrary lengths, thereby enhancing its applicability across diverse argumentative structures;

- algorithmic completeness: completeness is guaranteed through the incorporation of path admissibility constraints and a convergence condition, ensuring the robustness and reliability of the computational process;

- consistency in acceptability thresholds: 0 is maintained as both the acceptability threshold and the initial score for free arguments, providing a consistent baseline throughout the framework;

- convergence within a defined interval: acceptability scores for connected arguments converge within the interval $[-1, 1]$, with a direct correlation between the numerical score and the argument's acceptability—the closer the score is to 11, the higher the acceptability of the argument

- adherence to the fading principle: the influence of longer paths diminishes due to damping effects, in line with the fading principle, which reflects the decreasing impact of indirect support or attack over extended pathways.

Overall, this framework is designed to be general-purpose while emphasizing efficient utilization of computational resources. Such efficiency is particularly crucial for agents operating within a CPS, where time and resource constraints are a significant concern.

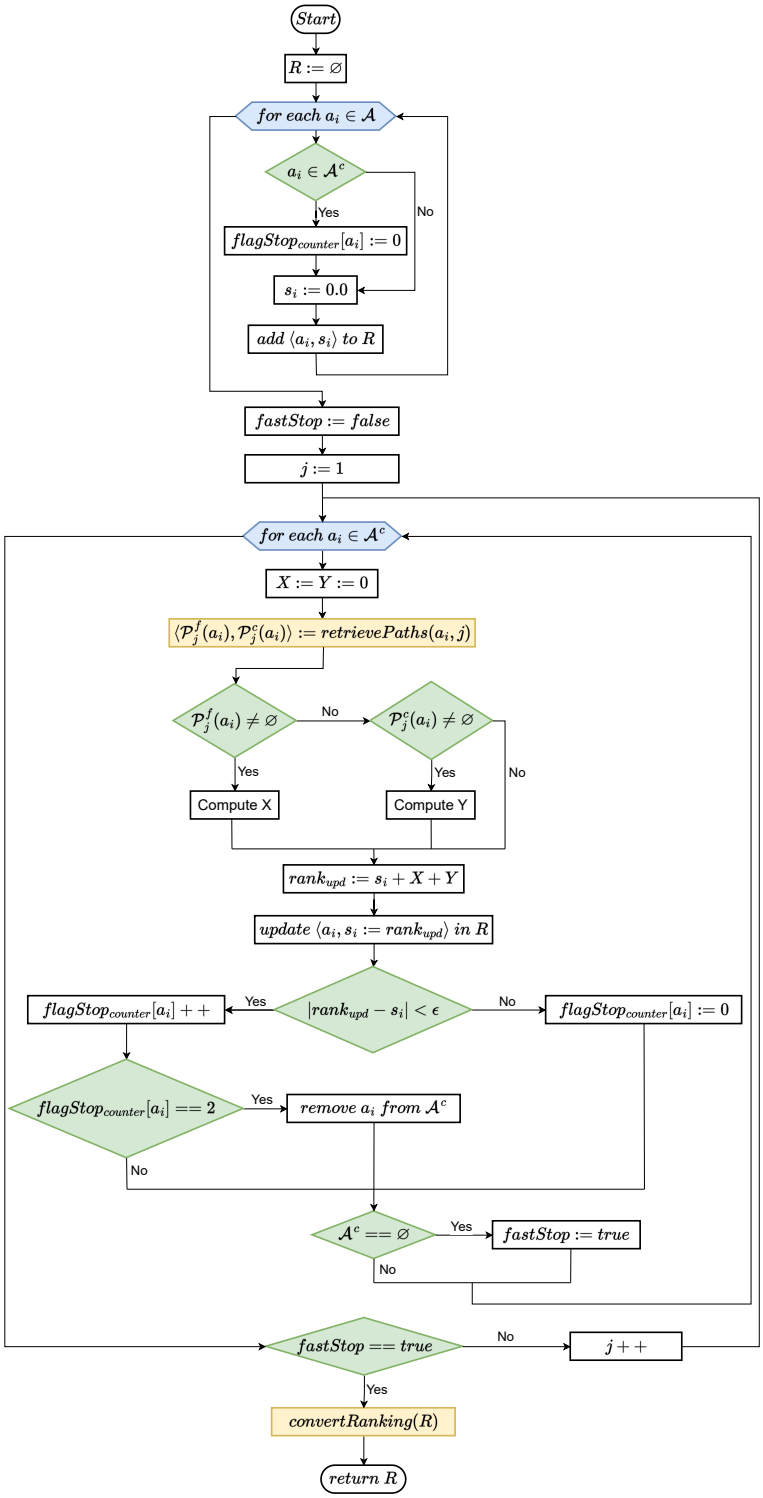


Figure 4.1: Relation type definition algorithm between pairs of arguments

Chapter 5

Case studies

In this chapter, a series of case studies for the proposed Knowledge–Based DLT infrastructure described in Chapter 3 are proposed. Specifically, the first two use cases illustrate distinct DLT infrastructures for events tracking in both public and private industrial context. The former addresses the steps involved in *e-procurement* processes within the supply chain, while the latter focuses on management, secure storing and standardization of supply chain documents. Furthermore, the following two use cases propose applications of semantic-based blockchain architectures in the field of *Smart Mobility*. Ridesharing services first and electric mobility charging infrastructures next are enhanced by means of formal and complete semantic description of available resources and user requests, allowing automated reasoning and, ultimately, user decision support.

These examples aim to assess the feasibility and practical applicability of the approaches and infrastructure components outlined in the previous chapters. In particular, each prototype illustrates how the paradigms introduced in this dissertation can bring useful elements of innovation and improvement in fields where the application of the adopted foundational technologies is already starting to be explored. Early implementations in these domains have highlighted some challenges, and the case studies presented here demonstrate how the proposed solutions can effectively address some of these issues, mainly in terms of robust traceability and accountability, adherence to standards, and autonomous and intelligent knowledge discovery and decision-making, thus representing a step further in maximizing the potential of these emerging technologies.

5.1 Public traceability of confidential processes in B2B marketplaces

E-procurement processes play a crucial role in modern supply chains, requiring strong security and transparency mechanisms to prevent fraud and ensure accountability. *Hyperledger Fabric*, an enterprise-grade permissioned distributed ledger platform, offers the foundational infrastructure to meet these requirements. The following use case for the DLT architecture described in Chapter 3 outlines the architectural design of an *e-procurement* system built on Hyperledger Fabric. A prototype has been implemented, where data and procedures involved in tendering workflows have been modeled.

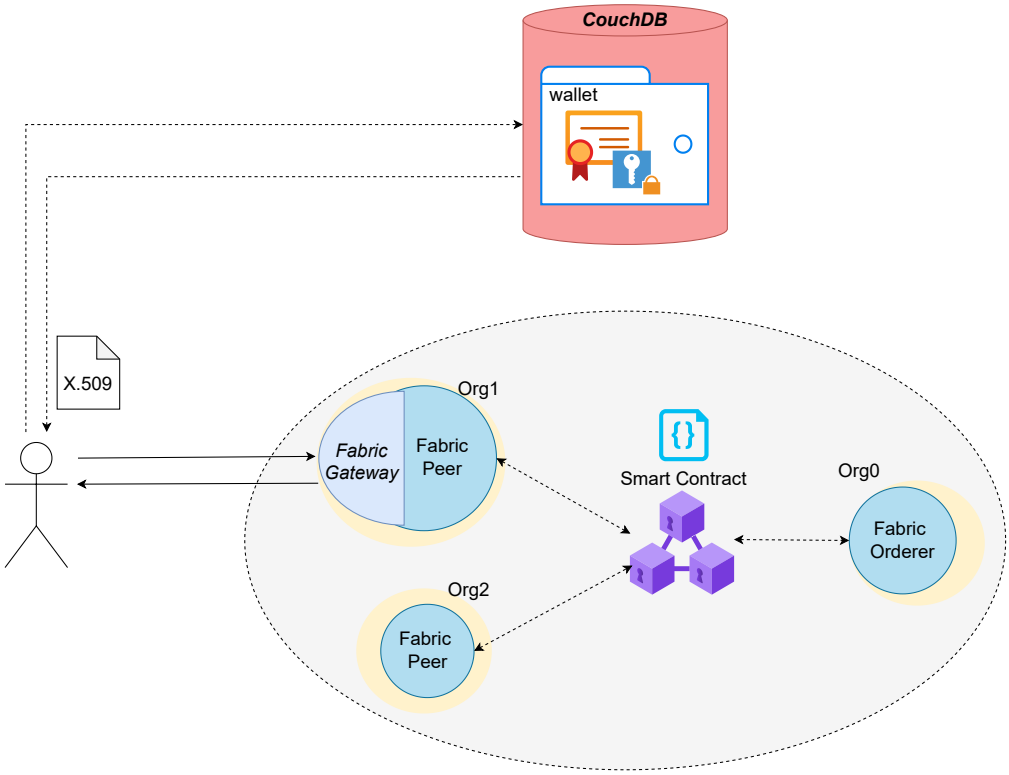


Figure 5.1: Architecture of the blockchain infrastructure based on Hyperledger Fabric

According to Fabric specifications, there are two types of nodes: *peer nodes*, that maintain the copies of the ledger and where SC are deployed as *chaincodes*, and *orderer nodes* responsible for ordering transactions into

blocks and distributing them to connected peers for validation and final commit to the ledger. To sign and validate transactions, each client must provide a signature using its private-public key pair.

As shown in Figure 5.1, the Fabric infrastructure is organized with three nodes, each managed by a single distinct organization for simplicity. They are structured as follows: (i) *Org1* manages a peer node that includes a *Fabric Gateway* component; (ii) *Org2* manages a second peer node; (iii) *Org0* manages an orderer node. The cryptographic material of users (*i.e.*, buyers and suppliers) interacting with the system, are stored in a deployed instance of a *CouchDB* NoSQL DataBase Management System (DBMS), which can be directly managed by the Fabric libraries. In detail, in the CouchDB instance, a database acts as a *wallet*, storing user certificates. Each certificate includes both the public and the private key of the user, with the private key encrypted using a symmetric key. For this purpose, the Fabric libraries have been modified to support automatic encryption and decryption of private keys every time they are stored or retrieved when needed.

Given these components, a *Client* exposes a set of endpoints in a ReST API that buyers or suppliers can use to interact with the blockchain infrastructure and invoke the corresponding SCs. This abstraction hides the internal complexities for SC invocation, transaction generation, validation, final commit and interaction with the DBMS. Moreover, by using the appropriate *Client* APIs, users can request their own certificate, in X.509 format, and private key from the DBMS before interacting with the system. Once obtained, these credentials are passed to the API that interacts with the *Fabric Gateway*[1] component, along with the necessary details for the specific SC invocation. From then on, the Fabric Gateway is in charge of handling the workflow that involves the SC invocation, with the necessary steps that include the modification of the state, the generation of the transaction, the arrangement into blocks by the ordering node and the final validation and commit by the peer nodes. Upon completion, the Gateway sends a response to the user, indicating the state changes that have been successfully executed.

[1] `https://hyperledger-fabric.readthedocs.io/en/latest/gateway.html`

5.1.1 Modeling of assets for tendering

To register all potential assets and related events of a tendering process on the blockchain, a specific data model has been defined using a series of *ProtocolBuffer*[2] messages. Each method of the SC relies on these data structures to store and modify data on the blockchain.

First of all, the generic asset registered on the blockchain consists of the following fields:

- *ID*: unique identifier;
- *created_by*: identifier of the user who created the asset (potentially derived from the certificate passed when invoking the specific SC);
- *type*: type of registered asset;
- *events*: a set of all events related to that asset within the system.

```
message Asset
{
    required string ID = 1;
    required string created_by = 2;
    required string type = 3;
    repeated Event events = 4;
}
```

Then, each individual event is recorded with the following fields:

- *event_ID*: identifier of the recorded event;
- *type*: type of recorded event;
- *created_by*: identifier of the user who generated the event;
- *timestamp*: timestamp of the event registration in the system.

```
message Event {
    required string event_id = 1;
    required string type = 2;
    required string created_by = 3;
    required int64 timestamp = 4;
}
```

[2] https://protobuf.dev/

This data structure is included in the ProtocolBuffer message that models a tendering process. It includes the following fields:

- an asset field of type *Asset*, including the list of events associated with the corresponding tender (*e.g.*, tender creation, field modifications, status changes, addition of communications or offers, addition of evaluations, etc.);
- a set of metadata with specific parameters related to the tender, such as status, subject, description, amount and type of offers that can be submitted.

```
message Bidding {
    required Asset asset = 1;
    required string state = 2;
    required string award_type = 3;
    required string offer_type = 4;
    required technical_score_weight = 5;
    required economical_score_weight = 6;
    [other metadata];
    repeated Offer offers = 16;
    repeated Message message = 17;
}
```

This data structure also includes all communications between the supplier and the buyer, stored in a nested data structure format, which includes the sender and the recipient, the subject of the message, its content, and the timestamp.

```
message Message {
    required string sender = 1;
    required string receiver = 2;
    required string object = 3;
    required string text = 4;
    required string timestamp = 5;
}
```

The tender also contains a field that collects all the submitted offers. Each offer is modeled as follows:

```
message Offer {
    required string createdBy = 1;
    required int32 amount = 2;
    optional Approval administrative_evaluation = 3;
    optional TechnicalEvaluation technical_evaluation = 4;
    optional EconomicalEvaluation economical_evaluation = 5;
}
```

Thus, in the blockchain, a tendering data structure is designed to track all offers submitted by suppliers, with fields that indicate the creator (*created_by*) and the specified offer amount (*amount* field). This data structure also allows the storage of administrative, technical, and economic evaluations recorded in the system before awarding the bidding, using the fields *administrative_evaluation*, *technical_evaluation*, and *economic_evaluation*, respectively.

5.1.2 Smart Contract implementation

Specific auction operations have been implemented within the SCs. Rather than handling generic assets and event, implementing the tendering process in a more granular way, enables a better validation on input field values. Conversely, registering only generic assets without additional details prevents the validation of corresponding values and limits the ability of the blockchain-based infrastructure to closely monitor the tendering process.

The main actors involved in the bidding processes are buyers and suppliers. The former is in charge of defining and opening the bidding process and has a complete view of all its steps, with the ability to also change the state of the process and give evaluations for offers before closing it. The suppliers are responsible for the insertion of offers and are authorized to inspect the public details of the process. The implemented methods and their respective validations are as follows:

- **Create**: creates a bidding data structure on the blockchain, including all values for the specified parameters and recording the creation event. After confirming that the auction does not already exist, the following fields are validated:

  - *award type*: must be either “Most Economically Advantageous Offer” or “Lowest Price”;
  - *offer type*: must be either “Fixed Amount” or “Percentage Discount” (on the initial price);
  - the sum of the fields related to the weight of the economic score and the technical score must equal 100;

- **UpdateField**: updates the auction fields recording a modification event, while the auction is in the preparation phase and not yet in progress. Validation ensures that the auction’s status is “In Preparation”, otherwise the transaction is rejected;

- **StateChange**: updates the auction’s state, according to the value of the new state passed as input. Valid states, in sequential order, are “In Preparation”, “In Progress”, “Under Evaluation” and “Closed”. If the auction is not in a state from which it can transition to the new subsequent state, the transaction is not applied;

- **InsertOffer**: adds an offer into the auction when its status is “In Progress”, recording the related event. Validation ensures the auction is actually “In Progress”;

- **InsertMessage**: inserts a message into an auction and records the event. Validation checks if the auction’s status is not “In Preparation”.

- **AddBiddingEvent**: inserts a generic event without metadata related to a specific auction.

When any of the SC methods defined here is invoked, the user’s certificate is passed along. Then, the user’s role is always verified by extracting the “role” field from the certificate. Based on this, the methods *Create*, *StateChange* and *UpdateField* can only be invoked by users with a “buyer” role. Additionally, this category of user can also invoke the *InsertMessage* method. Conversely, users with a “supplier” role can only invoke *InsertOffer* and *InsertMessage*. For the *AddBiddingEvent* method, which involves a generic event type, validation can be performed based on the event type to allow only specific roles to insert certain types of events.

5.2 Industrial CPS event tracking

For this use case, a DLT-based platform for managing document flow in supply chain processes has been designed and developed. The main features and functional requirements of the system are as follows:

- the platform must enable the storage and tracking of structured documents in *EPCIS 2.0* format;
- users must be able to upload documents produced by supply chain processes in formats different from the EPCIS standard, *e.g.*, supported by a third-party application. These documents will be internally converted to the *EPCIS 2.0* format;
- users must have the ability to retrieve documents stored in the system, and these documents must be returned in the original format supported by the third-party application that generated them, despite being stored in *EPCIS 2.0* format internally;
- documents must be stored in a DAG-type DLT. Specifically, the proposed system employs the IOTA distributed ledger;
- the platform must support the archival of generic EPCIS documents, allowing users to record events related to different business processes.

Moreover, since the *EPCIS* standard documents are internally generated by the platform from external data provided by a third-party application, it is assumed that users will have access to an additional front-end interface (whose implementation is not detailed here). This interface will perform client-side validation before actually transmitting data to the framework endpoints. Such an approach would also help to reduce system complexity and server load by offloading such tasks to external components.

Following these guidelines, the system is based on two main infrastructures, as shown in Figure 5.2:

- a service layer that includes two core components: the *nodejs-service* and the *rust-service*;
- the IOTA network, comprising the nodes that manage the Tangle data structure for document handling and storage.

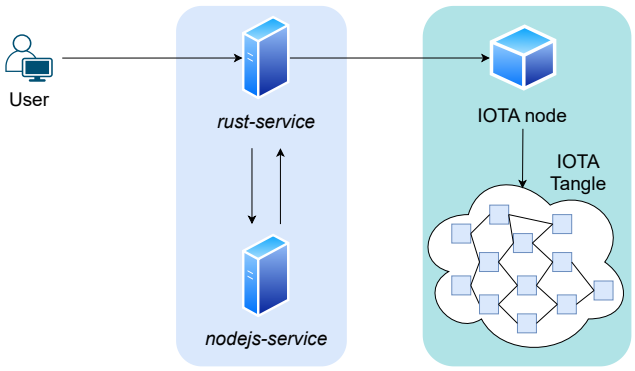


Figure 5.2: Architecture of the system

The *rust-service* component acts as the external access point for the platform, providing a ReST API through which users can access the main functionalities. In detail, the *rust-service* module consists of the following components:

- **controller**: defines the endpoints for handling requests from external applications and invokes the appropriate services defined by other components;
- **IOTA Client**: manages the integration of the document tracking system with the IOTA network. It utilizes the IOTA Streams library – already discussed in Chapter 3 – to create separate and private channels where only authorized parties can store and exchange data.

By interacting with the endpoints of the *rust-service* API, users can request data insertion into the system or read the stored information (Figure 5.3).

Before the data is inserted into the Tangle, a data pre-processing phase is performed via interaction with the *nodejs-service*. In detail, as depicted in Figure 5.3(a), whenever the *rust-service* controller receives a request to insert a message, the content is passed to the *nodejs-service* through a request to the HTTP POST endpoint `/toEPCISDocument`. At this point, the *nodejs-service* initiates an internal task to convert the document into the *EPCIS* standard using appropriate JSON mapper files. The translation result is encapsulated in the response of the POST request and sent back to the controller. Then, the internal IOTA client service is invoked to publish the message to the DAG-based distributed ledger.

	Insertion	**Deletion**	**Observation**
EPCIS event type	Object	Object	Object
Action	ADD	DELETE	OBSERVE
Other fields	EventId	EventId	EventId
Fields (what)	epcList quantityList	epcList quantityList	epcList quantityList
Fields (when)	EventTime EventTimeZoneOffset	EventTime EventTimeZoneOffset	EventTime EventTimeZoneOffset
Fields (where)	bizLocation	bizLocation	bizLocation
Fields (why)	bizStep disposition	bizStep disposition	bizStep disposition

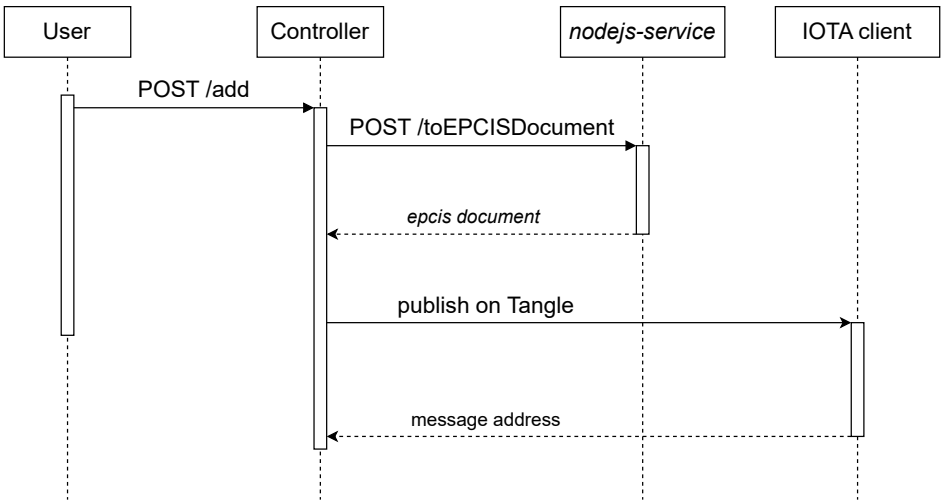
Table 5.1: Modelled EPCIS events

Similarly in Figure 5.3(b), when a user sends a request to read a previously stored message, the controller passes the information transmitted in the request to the IOTA client, which retrieves the relevant messages from the Tangle. This is returned to the controller, which then forwards the data to the `/fromEPCISDocument` endpoint of the *nodejs-service*. The *nodejs-service* converts the message into the format supported by the third-party platform and encapsulates the result in the response. The controller, in turn, includes this document in the body of the HTTP response returned to the user.

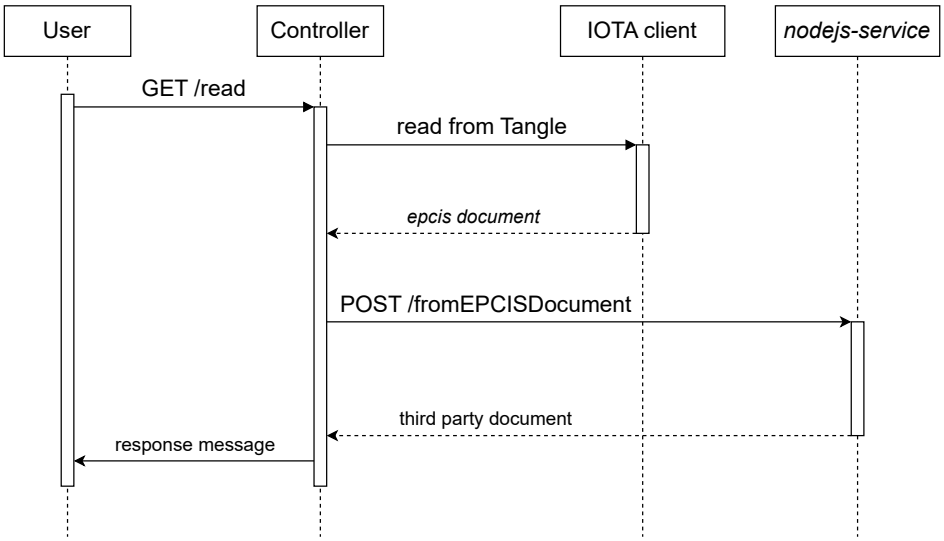
5.2.1 Definition of supported EPCIS documents

To enable the insertion of events related to various business processes, a specific structure for EPCIS documents has been defined, outlining the types of events the platform can handle. For this purpose, only a relevant subset of the original set of EPCIS events has been considered. The structure of the events taken into consideration is detailed in Table 5.1.

All events managed by the platform are classified as `ObjectEvent`, representing generic events. The `Action` field is responsible for identifying the type of EPCIS document to be archived and the events to be tracked. `ObjectEvent` events are characterized by the mandatory `EventId` field, which uniquely identifies an event. The remaining fields define the structure of documents supported by the system and can be categorized into four dimensions, *What*, *When*, *Where*, and *Why*, as follows:



(a)



(b)

Figure 5.3: Sequence diagrams showing (a) the write operation and (b) the read operation.

- *What* dimension, with the fields:
  - `epcList`: uniquely represents an object;
  - `quantityList`: indicates the quantity and class of objects specified in *epcList*;
- *When* dimension, with:
  - `EventTime`: date and time of the event creation;
  - `EventTimeZoneOffset`: time zone of the event creation location;
- *Where*:
  - `bizLocation`: the business location associated with the event;
- *Why*:
  - `bizStep`: the business process step in which the event takes place;
  - `disposition`: the current state of the objects involved in the event.

The *nodejs-service* component is responsible for translating documents from the format used by third-party platforms to the *EPCIS 2.0* standard document structure, and viceversa. Its architecture is illustrated in Figure 5.4. For this purpose, two endpoints have been defined:

- `/toEPCISDocument`: a POST HTTP request. The request body contains data in `application/json` format, corresponding to the format generated by an external application. The returned response is a JSON document compliant with the EPCIS 2.0 standard;
- `/fromEPCISDocument`: similar to the previous one, it is a POST request that accepts `application/json` content in EPCIS 2.0 format and returns a response in a format related to that of an external application.

Internally, the *Document Generator* component handles the translation of the input document into the *EPCIS 2.0* data structure. This translation is accomplished using two *JSON* files:

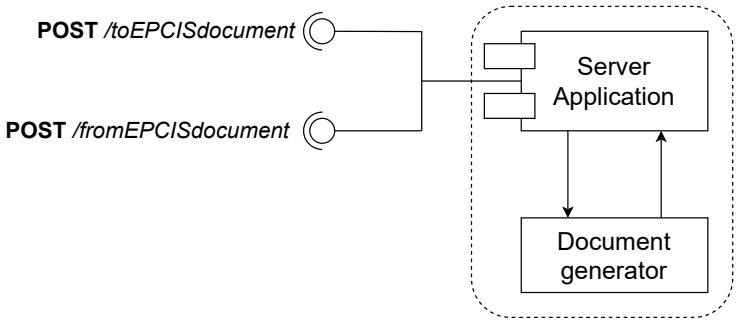


Figure 5.4: Structure of the *nodejs-service*

- `mapperJSONtoEPCIS.json`: maintains a key-value mapping where the keys represent field names from the document provided by the third-party application, and the respective values correspond to the field names compliant with the EPCIS standard;
- `mapperEPCIStoJSON.json`: defines the key-value mapping between EPCIS fields and their corresponding proprietary format fields.

5.2.2 Document tracking on the IOTA Tangle

The structure of the *rust-service* component is depicted in Figure 5.5.

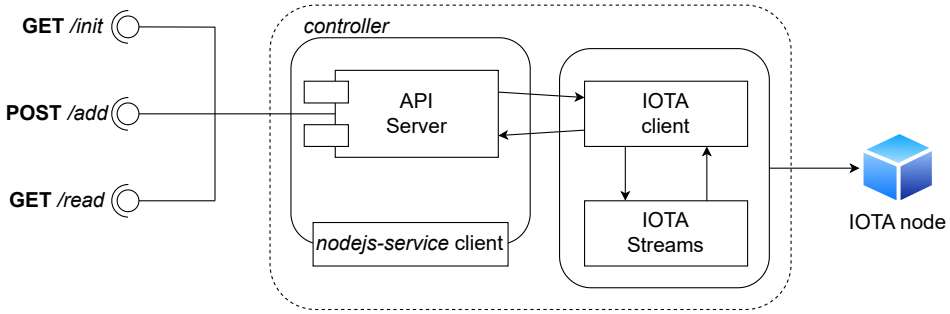


Figure 5.5: Structure of the *rust-service* component

The component named *Controller* manages various workflows by interacting with specific services at appropriate times. It incorporates two services:

- *API server*: exposes endpoints for requests from proprietary applications;

- *nodejs-service client*: an HTTP client used to requests document translation from the *nodejs-service*.

The *API server* provides three ReST endpoints:

- `/init`: an HTTP GET request. Input parameters include the number of IOTA Streams channels and subscribers to initiate. The platform manager makes an *init* request specifying the number of channels and subscribers expected in the Streams protocol. Upon successful completion of channel and subscriber creation, a confirmation message is returned;

- `/add`: an HTTP POST request that archives a document in the system. The document to be stored is included in the request body along with two parameters: `author_seed`, which determines the channel on which the document will be published, and `message_id`, a public identifier of the sent message. In response to the add request, the user will receive a confirmation message acknowledging receipt of the document;

- `/read`: a GET request to retrieve documents from the system. It requires three parameters: `subscriber_seed`, which identifies the subscriber fetching the message, `channel`, publicly identifying the IOTA channel from which the message will be extracted, and `message_id`, which specifies the packet containing the requested document. The response contains the requested document in the format supported by the third-party application.

The *IOTA node*, depicted in Figure 5.5 represents the connection point between the *rust-service* component and the IOTA network. This node can be deployed as a peer connecting to one of the official IOTA networks such as *Devnet*, *e.g.*, the *Chrysalis Devnet* or *Mainnet*, or a network managing a private Tangle.

The *IOTA client*, responsible for interacting with the Tangle, is defined by two classes, `TangleReader` and `TangleWriter` (Figure 5.6), to manage connection to *IOTA Streams* channels.

`TangleWriter` objects are structures defined by:

- an author of type `User<Client>` that defines the channel owner used for publishing messages;

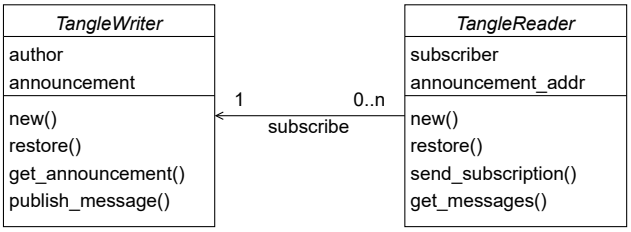


Figure 5.6: Class diagram for the *rust-service* component

- an `announcement` of type `Address` that corresponds to the *announcement link*, *i.e.*, the address of the announcement message of the channel created by the author.

In detail, when defining a `TangleWriter` object, the `TangleWriter::new()` method is called. This is invoked following an `/init` request to instantiate the number of authors required. To instantiate an author and its associated channel, the transport mechanism must be defined, *i.e.*, an object of type `Client` that establishes the means for sending and receiving binary messages from an IOTA network node, thus interacting with the Tangle. Two important configuration parameters passed for instantiating the `Client` object are:

- `node_url`: a mandatory value corresponding to the URL endpoint of the node part of an IOTA network, used by the client to read from and write to the Tangle;

- `with_local_pow`: it specifies whether the computation related to PoW as an anti-spam mechanism is to be performed locally by the client or directly on the node. When setting `with_local_pow` to false, the client transmits the message directly to the specified `node_url`, and if this peer has local PoW enabled in its configuration, it will execute it on the received packet before publishing it on the Tangle.

Once the author is defined, the channel, of which the author will be the owner, is created. This operation is performed by calling the `create_streams()` function, which returns the announcement message of the channel to which subsequent message traffic will be linked. Furthermore, the `restore()` method is invoked for this purpose, which is very similar to the `new()` method described above. Specifically, once a transport layer is defined, it

retrieves a previously instantiated author, based on two needed parameters:

- `seed`: a character string used to instantiate a user-type client;
- `address`: the address of the announcement message published when the channel was initiated.

5.3 Smart mobility services

DLTs have showed several benefits across a wide variety of domains and they can prove particularly interesting and suitable for innovative use cases in the field of intelligent Mobility-as-a-Service (MaaS) infrastructures. One key advantage is that blockchain can guarantee transparency and trust in transportation systems, providing travellers with reliable access to immutable information. Furthermore, the process of tokenizing and tracking ticketing data, monetary transactions for service exchange, and personal information has effectively demonstrated the potential to increase efficiency while ensuring secure and reliable communications, particularly for Internet of Vehicles (IoV) applications.

Blockchain-based platforms have been applied in areas such as smart car parking services [140], car leasing [67], autonomous cars training [131], as well as the establishment of trusted multiparty insurance arrangements [103]. Furthermore, these platforms have been employed for securing transactions involving the buying and selling of used cars, as well as for transparent and trusted dissemination of the usage history of motors for trading purposes [117]. In [11] a smart contract-based platform dedicated to emerging transport services was proposed, allowing data about drivers and vehicles profiles and privileges acquired after the completion of a service to be stored and shared. To protect data stored on the limited pool of resources of smart vehicles, a secure content caching scheme was developed, employing a private blockchain alongside a *Deep Reinforcement Learning* (DRL) approach [12].

The architecture proposed in [79] leverages blockchain inherent transparency and trust to enable direct interactions between transport providers and travellers, eliminating the need for intermediaries. Travellers request routes and receive information on different transportation modes. Based

on their preferences, they invoke a SC and transport providers execute it to compare the request with their route data. Once verified, the SC data is added to the blockchain. Scalability is addressed distributing the computation on multiple blockchains, each managed by local transport providers. The hierarchy of blockchains allows providers to join relevant regional blockchains without storing extensive data. However, challenges arise in connecting different blockchains and transferring route information across in a consistent way.

The proposal in [48] focuses on the complex issue of billing in multimodal transport services involving various providers. To adress this, the platform charges users based on actual usage via the MIOTA cryptocurrency. Vehicles publish their offers to the IOTA Tangle, enabling users to browse and manage service details and payments directly with the vehicle, independent of the blockchain. A user can browse mobility offers at a location, choose a vehicle, and connect to it through an app. The transaction takes place through an IOTA micropayments channel, allowing flexible and precise travel. At the end of the journey, the channel automatically closes, and the vehicle becomes available for other users. The vehicle registration process is minimal, requiring only a check-in message on the IOTA address of the location.

5.3.1 Blockchain-based ridesharing services enhanced with semantics

In the context of *Smart Mobiliy* and MaaS, a prototypical carpooling service named *RideMATCHain*[3] has been developed to underline the benefits of the semantic-based microservice framework described in Section 3.4.

Let us suppose a heterogeneous group of people is sending requests from different places and at various times to a ridesharing service in order to reach their destinations. For example, let us consider two drivers currently available for ridesharing through the service with their cars. The problem consists in matching every passenger with the car better fulfilling their requirements, considering user preferences and compatibility issues. All passengers and vehicles are described according to a prototypical ontology, whose taxonomy of concepts is depicted in Figure 5.7. Whenever possi-

[3]GitHub repository: `https://github.com/sisinflab-swot/ridematchain`

ble, classes have been reused from *SUMO-OWL* (the OWL translation of the Suggested Upper Merged Ontology) [69], *e.g.*, `Automobile`, `AgeGroup`. Each available ride is a resource, represented as an asset stored into the blockchain with a semantic annotation split into two parts: the first one is constant over time and summarizes the set of vehicle features relevant to the service and the driver's profile; the second part specifies information about available space for passengers and their luggage.

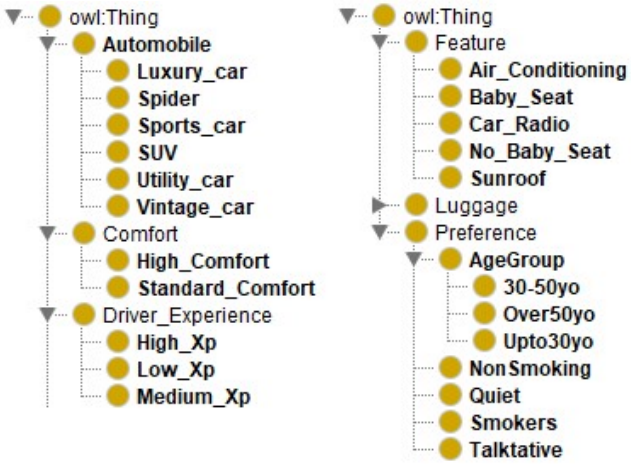


Figure 5.7: Reference ontology top-level classes

An excerpt of the annotations linked to each car is shown below in Manchester syntax [41]:

```
SUV1:  SUV and (accepts only NonSmoking) and (has_Feature only (Car_Radio
and Air_Conditioning and Baby_Seat)) and (comfort_Level only High_Comfort)
and (driver_Experience only High_Xp) and (available_Seats exactly 6)
and (available_Capacity exactly 650)

Small_CityCar1:  Utility_Car and (has_Feature only (Sunroof and
No_Baby_Seat)) and (comfort_Level only Standard_Comfort) and
(driver_Experience only High_Xp) and (available_Seats exactly 1) and
(available_Capacity exactly 200)
```

Resource descriptions have been registered, distributed and synchronized as assets among all participants in the blockchain, by means of the Registration SC described in Section 3.4. For the sake of simplicity, to show how the ridesharing service works, let us consider a single step iteration. *A user, headed downtown with her baby, is looking for a car with two seats,*

one of which equipped with baby seat. She would like to travel comfortably in an air-conditioned, quiet and non-smoking environment. In formulas:

```
R ≡ (accepts only (NonSmoking and Quiet)) and (has_Feature only
(Baby_Seat and Air_Conditioning)) and (comfort_Level only High_Comfort)
and (driver_Experience only High_Xp) and (available_Seats min 2)
```

In order to take into account passenger-car and passenger-passenger constraints, the framework invokes the `Discovery SC`, which in turn triggers the semantic matchmaking task described in Section 2.2.3 and detailed in [106]. After a pre-filtering step, exploited to discard cars with a heading diverging from that of the passenger, the matchmaking finds which vehicles of the fleet best meet the specified requirements. The returned list of resources is ranked according to the following *utility function*:

$$u(R,C) = 100[1 - \frac{s\_penalty(R,C)}{s\_penalty(R,\top)}(1 + \frac{distance(R,C)}{max\_distance})]$$

where $s\_penalty(R,C)$ is the semantic distance between passenger profile R and car annotation C; this value is normalized dividing by $s\_penalty(R,\top)$, which is the distance between R and the *universal concept* $\top$ (a.k.a. *Top* or *Thing*) and depends only on the ontology structure. The $max\_distance$ contextual data is used to take into account the car geographical distance, combined as weighting factor. The purpose of the utility function is also to convert the semantic distance score to a more user-friendly [0, 100] ascending scale. It is important to notice that in case of a full match $s\_penalty(R,C) = 0$ hence $u(R,C) = 100$ regardless of distance. Outcomes, ranked according to the utility function, are 90.2 and 76.2 for `SUV1` and `Small_CityCar1`, respectively.

As evident, SUV and passenger semantic descriptions are very similar: all atomic concepts, universal quantifiers and unqualified number restrictions on roles are compatible. Conversely, passenger requirements are not compatible with the small city car: the user can get an explanation for this from the `Explanation SC` by means of the Concept Contraction reasoning task:

```
G: (has_Feature only (Baby_Seat)) and (available_Seats min 2)
```

The best match for the passenger is with `SUV1`, so she is assigned to it. When the passenger/car association is confirmed, the passenger's preferences (modelled in the filler of the `accepts` object property) are appended in conjunction with the SUV annotation. This task is performed by the

`Selection SC` that enables the interaction with the selected resource and updates its semantic description (modifications are underlined):

`SUV1: SUV and (accepts only (NonSmoking and Quiet)) and (has_Feature only (Car_Radio and Air_Conditioning and Baby_Seat)) and (comfort_Level only High_Comfort) and (driver_Experience only High_Xp) and (available_Seats exactly 4) and (available_Capacity exactly 350)`

The selection transaction allows renewing the blockchain internal status and repeating the allocation task for the remaining passengers, verifying any incompatibility constraints.

5.3.2 CPS-oriented blockchain for green mobility

A case study on the power management of PEVs in a Smart Grid has been developed to validate capabilities of the proposed framework in dynamic CPS scenarios. The illustrative example discussed hereafter is provided to clarify the proposal.

A smart PEV V_1 informs its driver that battery level is below 50%. The driver confirms that charging is needed. Blockchain assets available are charging slots offered in parking areas. High-level descriptive features of chargers are annotated w.r.t. a domain ontology (not reported due to dearth of space). Data-oriented attributes (*e.g.*, latitude and longitude of parking areas) are also included. The providers register chargers by invoking the Registration SC (described in 3.4) through smart parking area WI node.

Since V_1*'s battery level is not critical, it looks for the best available charging service in the area.* The vehicle acts as a Consumer on the blockchain, requesting the execution of a Discovery SC in *full* mode. Figure 5.8 reports on V_1's charge request. For the sake of compactness, concept expressions are simplified w.r.t. the ones actually used for the case study. In particular, for each universal restriction of the form `role only concept`, the reader should assume there is a corresponding `role some owl:Thing` existential restriction in conjunction. Reported matchmaking results refer to the complete expressions. in OWL 2 *Manchester syntax* [134]: charging rate of at least 25 km per charging hour, minimum available energy of 12 kWh at charging station, maximum distance of 50 km between charging station and energy dispatcher, and maximum fee of 50 cents per kWh are requested. Car location and maximum distance between vehicle and charg-

```
ChargeRequest EquivalentTo: V1 and (has_Charging_Rate_Per_Hour only (km
min 25)) and (has_Available_Power only (kWh min 12)) and
(has_Dispatcher_Distance only (km max 50)) and (has_Base_Fee_Per_kWh only
(cents max 50))

V1 EquivalentTo: Electric_Vehicle and (has_Output_Rate only (daW max
37)) and (has_Current only (Ampere max 16)) and (has_Voltage only (Volt
max 230)) and (has_AC_Compatible_Plug only VDE-AR-E_2623-2-2Type2_Plug)
and (has_DC_Compatible_Plug only CCSCOMBO2_Plug)
```

Figure 5.8: Semantic annotation of vehicle request

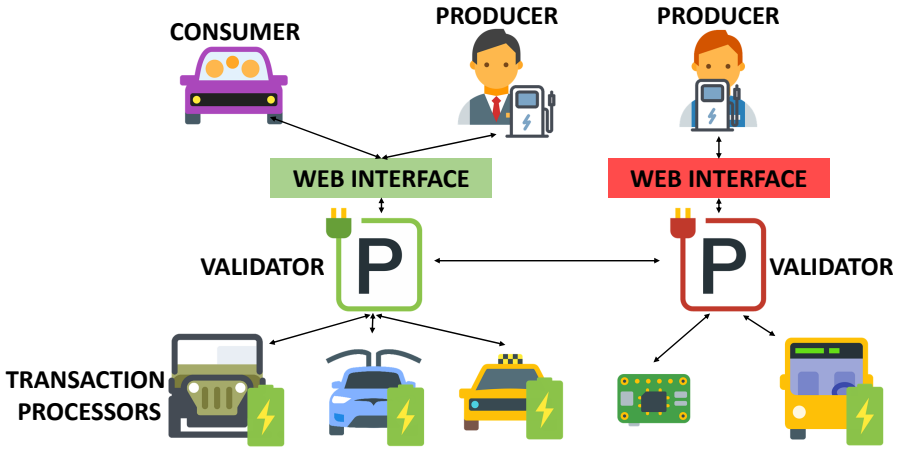


Figure 5.9: Roles in the Smart Mobility case study.

ing station are specified as data-oriented request attributes. The request also defines the start time and duration of the charging service the vehicle is willing to reserve. The vehicle model adopts a standard *VDE-AR-E 2623-2-2* (Type 2) connector and receptacle for AC charging, incompatible with *SAEJ1772* (Type 1). Mutually incompatible *CHAdeMO*, *CCS Combo1* or *CCS Combo2* connectors exist for fast DC charging. Vehicle V_1 supports CCS Combo2 plug only.

The Discovery request is divided in pieces through the Web Interface and forwarded to a Validator. In the reference scenario, parking areas play the role of Validators because they have no mobility issues or energy supply limits, so they can run the consensus protocol reliably. The smart vehicles parked or in charging within parking areas work as Transaction Processors, since they have enough on-board computing capabilities; they receive a compensation for their work in the form of savings on current or future charging fees. However, as Figure 5.9 highlights, heterogeneous devices –even embedded ones– can be also exploited as Transaction Processors,

```
ChargingStationA_Slot1 EquivalentTo: (has_AC_Compatible_Plug only
SAEJ1772-2009Type1_Plug) and (has_EVSE_Profile only AC_Level_2) and
(has_Available_Power only (kWh exactly 500)) and (has_Base_Fee_Per_kWh
only (cents exactly 55)) and (has_Charging_Rate_Per_Hour only (km exactly
45)) and (has_Dispatcher_Distance only (km exactly 60))

ChargingStationA_Slot2 EquivalentTo: (has_DC_Compatible_Plug only
CCSCOMBO2_Plug) and (has_EVSE_Profile only DC_Level_1) and
(has_Available_Power only (kWh exactly 500)) and (has_Base_Fee_Per_kWh
only (cents exactly 85)) and (has_Charging_Rate_Per_Hour only (km exactly
200)) and (has_Dispatcher_Distance only (km exactly 60))

BuildingB_Slot1 EquivalentTo: (has_AC_Compatible_Plug only
VDE-AR-E_2623-2-2Type2_Plug) and (has_EVSE_Profile only AC_Level_1) and
(has_Available_Power only (kWh exactly 100)) and (has_Base_Fee_Per_kWh
only (cents exactly 45)) and (has_Charging_Rate_Per_Hour only (km exactly
20)) and (has_Dispatcher_Distance only (km exactly 25))
```

Figure 5.10: Semantic annotations of available resources.

thanks to the multiplatform IoT-oriented support of Sawtooth. The Discovery SC pre-filters available resources on location and temporal context attributes: resources outside the query area or unavailable at the specified time are discarded. The chain resources satisfying the above constraints are shown in Figure 5.10. Full discovery based on semantic-matchmaking returns a ranked list of the three resources w.r.t. the request, where *BuildingB_Slot1* is the resource having the highest semantic affinity score, despite the low charging rate per hour. In detail, the overall match score s for resource S_i is computed by means of the formula:

$$s(S_i) = \begin{cases} 1 - p(S_i), & 1 - p(S_i) > s_{min} \\ 0, & otherwise \end{cases} \tag{5.1}$$

with the semantic penalty function p computed as:

$$p(S_i) = \frac{w \cdot penalty_c(S_i) + (1 - w) \cdot penalty_a(S_i)}{penalty_{max}} \tag{5.2}$$

where $penalty_c$ is the penalty calculated by Concept Contraction between the request R and each resource annotation S_i, while $penalty_a$ is the penalty value of the Concept Abduction procedure between the consistent part K of the request R and S_i. The value of p is normalized w.r.t. $penalty_{max}$, *i.e.*, the maximum possible semantic distance (which is the one between R and the most generic concept $\top$, and depends only on axioms in the reference

domain ontology [105]). The parameter w can be set in the $]0, 1[$ interval and it determines the relative weight of explicitly conflicting elements in S_i w.r.t. R; for the proposed case study $w = 0.8$ has been selected after preliminary tests.

ChargingStationA_Slot1 and *ChargingStationA_Slot2* are both too expensive and too far, with *ChargingStationA_Slot2* being the most expensive one. *ChargingStationA_Slot1* also equips an incompatible charging plug. V_1 has the option to retrieve motivations for the discovery result by means of the Explanation SC. Then the Selection SC allows selecting the best service: V_1 *pays the charging fee and reserves its use for the booked time.*

Chapter 6

Conclusions and perspectives

This thesis has addressed the main limitations of DLT infrastructures in terms of interoperability and scalability, that hinder their applicability in the CPS domain for effective multi-agent coordination. For this purpose, an innovative architectural solution has been proposed to attempt to tackle those issues by integrating different types of blockchain and DLT platforms, leveraging the typical micro-service architectural approach of IoT and MEC scenarios. Furthermore, a semantic-based approach has been integrated to enhance coordination in complex CPS systems, enabling resource-constrained devices to leverage processes of *discovery*, *negotiation* and *composition* of available resources and services, annotated as information fragments in standard OWL language. The overall direction is to create an intelligent MAS environment in which devices can act as smart context-aware agents capable of not only collecting information, but also exchanging data and engage in articulate dialogues. The procedures have all been grounded onto a decentralization and security substrate granted by a DLT platform, leveraging SCs. Unifying KRR and DLT has demonstrated to improve immutability, transparency and, ultimately, guarantee a high degree of explainability of all decisions taken by coordinating agents in a CPS.

A novel decentralized dual-layer architecture has been introduced to enable different traditional blockchain platforms to offer services and provide resources in a collaborative way. In this infrastructure, a layer called *L1* serves as the main entry point for users to interact with the system and leverages the *IOTA Tangle* to trace all interactions. IOTA has been

selected due to its lightweight nature and high transaction throughput, advantages that makes it suitable for scenarios where a high responsiveness is needed to coordinate several entities. A *message broker* enables a *publish-subscribe* model for bidirectional communication between the public layer *L1* and the private layer *L2*. The latter can be considered as a federated blockchain layer, where data and processing can be distributed on a set of heterogeneous conventional blockchain platforms. In this architecture, a request submitted by a user can be executed in a distributed manner involving all traditional blockchain platforms that are able to execute the requested service: this eases the computational burden w.r.t. to an approach in which a single blockchain platform is dedicated to the execution of a given set of service. Moreover, the presence of different blockchains on *L2* can contribute to improve privacy and separation of concerns. Data from multiple domains can be managed in the system by distributing it to dedicated *L2* blockchains. This enables a more granular control on available data, which is delegated to the traditional blockchain platforms that manage it, rather than relying on a single platform to define privacy policies regarding the whole dataset. Overall, each component has a specific function and the modular design can enhance scalability, since the addition of new components requires little or no modification to other parts of the system. This infrastructure has been integrated with a semantic-enhanced blockchain layer to allow the execution of distributed *discovery*, *negotiation* and *composition* procedures. In addition, a semantic-based argumentation framework has been investigated, enabling agents to interpret data not only using Semantic Web technologies, but also rely on *computational argumentation* for managing conflicts and reach consensus. This framework defines a *Bipolar Weighted Argumentation Framework* (BWAF) that applies DL-based semantic matchmaking to evaluate arguments relations. It categorizes and weights attacks and supports, building the argumentation graph, then exploits it to rank acceptability scores for arguments. The proposal caters particularly to resource-constrained environments, resulting in a computationally efficient, general-purpose solution that enables agents to autonomously make informed, justifiable decisions.

Future work includes a more torough experimental validation of the dual-layer DLT architecture, integrating both domain specific and semantic-based services, enabling the shift to a more scalable version of the *Seesaw*

prototype. This can lead to a more extensive and detailed validation of the proposed solution, as well as a test campaign to assess its performance and sustainability in different scenarios, such as the ones explored in developed case studies, *i.e.*, B2B processes, Smart Mobility and tracking of industrial CPS events. Moreover, the application of advanced semantic-based techniques of knowledge fusion among knowledge scattered on different blockchains on L2 can be investigated.

A more complete version of the two-layer DLT infrastructure may also leverage the IOTA substrate to manage identities using the *Decentralized Identifiers* (DIDs) standard [116], allowing all subjects interacting with L1 to use verifiable and shared identities. Additional improvements may involve embedding all logic regarding the management of communication between L1 and L2 into *IOTA Smart Contracts* (ISC). With this improvement, as soon as the user inserts messages in the IOTA Tangle, the related Smart Contract produces an Event which, in turn, triggers the appropriate notification on the Gateway, which is then ready to read from the Tangle, thereby eliminating the need for a Message Broker, which remains as a centralized component limiting the current proposal. For what concerns the argumentative BWAF framework, future work will aim to embed semantic explanations of relations into the argumentation graph in order to further improve the explainability of the framework. Moreover, a network of blockchain platforms can be leveraged to manage a larger graph, enabling a network of cooperating agents to discuss and dialogue and execute the ranking semantics algorithm in a decentralized way.

Bibliography

[1] Kazi Masudul Alam, Mukesh Saini, and Abdulmotaleb El Saddik. Toward social internet of vehicles: Concept, architecture, and applications. *IEEE Access*, 3:343–357, 2015.

[2] Leila Amgoud and Jonathan Ben-Naim. Ranking-based semantics for argumentation frameworks. In *International Conference on Scalable Uncertainty Management*, pages 134–147. Springer, 2013.

[3] Leila Amgoud and Jonathan Ben-Naim. Evaluation of arguments in weighted bipolar graphs. *International Journal of Approximate Reasoning*, 99:39–55, 2018.

[4] Farhan Amin, Abdul Majeed, Abdul Mateen, Rashid Abbasi, and Seong Oun Hwang. A systematic survey on the recent advancements in the Social Internet of Things. *IEEE Access*, 10:63867–63884, 2022.

[5] Luigi Atzori, Antonio Iera, and Giacomo Morabito. From "smart objects" to "social objects": The next evolutionary step of the internet of things. *IEEE Communications Magazine*, 52(1):97–105, 2014.

[6] Luigi Atzori, Antonio Iera, and Giacomo Morabito. Understanding the Internet of Things: definition, potentials, and societal role of a fast evolving paradigm. *Ad Hoc Networks*, 56:122–140, 2017.

[7] Luigi Atzori, Antonio Iera, Giacomo Morabito, and Michele Nitti. The Social Internet of Things (SIoT) – When Social Networks Meet the Internet of Things: Concept, Architecture and Network Characterization. *Computer networks*, 56(16):3594–3608, 2012.

[8] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele

Nardi, and Peter Patel-Schneider. *The Description Logic Handbook*. Cambridge University Press, 2002.

[9] Pietro Baroni, Antonio Rago, and Francesca Toni. How many properties do we need for gradual argumentation? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[10] Taher Abouzaid Abdel Aty Abdel Bary, Basem Mohamed Elomda, and Hesham Ahmed Hassan. Multiple Layer Public Blockchain Approach for Internet of Things (IoT) Systems. *IEEE Access*, 12:56431–56438, 2024.

[11] Christopher Bayliss. Machine learning based simulation optimisation for urban routing problems. *Applied Soft Computing*, 105:107269, 03 2021.

[12] Asma Belhadi, Youcef Djenouri, Gautam Srivastava, Djamel Djenouri, Alberto Cano, and Chun-Wei Lin. A two-phase anomaly detection model for secure intelligent transportation ride-hailing trajectories. *IEEE Transactions on Intelligent Transportation Systems*, 22:4496–4506, 09 2020.

[13] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic Web. *Scientific American*, 284(5):28–37, 2001.

[14] Tim Berners-Lee, Roy Fielding, and Larry Masinter. Uniform Resource Identifiers. RFC 3986, Internet Engineering Task Force, January 2005. `https://rfc-editor.org/rfc/rfc3986.txt`.

[15] Philippe Besnard, Alejandro Garcia, Anthony Hunter, Sanjay Modgil, Henry Prakken, Guillermo Simari, and Francesca Toni. Introduction to structured argumentation. *Argument & Computation*, 5(1):1–4, 2014.

[16] Nicola Bicocchi, Giacomo Cabri, Federica Mandreoli, and Massimo Mecella. Dealing with data and software interoperability issues in digital factories. In *Proceedings of the 25th ISPE Inc. International Conference on Transdisciplinary Engineering*, volume 7, page 13. IOS Press, 2018.

[17] Bluetooth. http://www.bluetooth.com.

[18] Elise Bonzon, Jérôme Delobelle, Sébastien Konieczny, and Nicolas Maudet. A parametrized ranking-based semantics compatible with persuasion principles. *Argument & Computation*, (12):49–85, 2021.

[19] Alexander Borgida. Description logics in data management. *IEEE transactions on knowledge and data engineering*, 7(5):671–682, 1995.

[20] Amel Bouzeghoub, Said Jabbour, Yue Ma, and Badran Raddaoui. Handling conflicts in uncertain ontologies using deductive argumentation. In *International Conference on Web Intelligence*, pages 65–72. ACM, 2017.

[21] Claudette Cayrol and Marie-Christine Lagasquie-Schiex. On the acceptability of arguments in bipolar argumentation frameworks. In *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty, ECSQARU*, volume 3571 of *Lecture Notes in Computer Science*, pages 378–389. Springer, 2005.

[22] J. Chamberlain, C. Blanchard, S. Burlingame, S. Chandramohan andE. Forestier, G. Griffith, M.L. Mazzara, S. Musti, S.-Y.Son, G. Stump, and C. Weiss. *IBM WebSphere RFID Handbook: A Solution Guide*. IBM International Technical Support Organization, May 2006.

[23] Ray Chen, Fenye Bao, and Jia Guo. Trust-based Service Management for Social Internet of Things Systems. *IEEE Transactions on Dependable and Secure Computing*, 13(6):684–696, 2016.

[24] Yong Chen, Yang Lu, Larisa Bulysheva, and Mikhail Yu Kataev. Applications of blockchain in Industry 4.0: a review. *Information Systems Frontiers*, pages 1–15, 2022.

[25] Zhikui Chen, Ruochuan Ling, Chung-Ming Huang, and Xu Zhu. A scheme of access service recommendation for the Social Internet of Things. *International Journal of Communication Systems*, 29(4):694–706, 2016.

[26] Konstantinos Christidis and Michael Devetsikiotis. Blockchains and Smart Contracts for the Internet of Things. *IEEE Access*, 4:2292–2303, 2016.

[27] Connectivity Standards Alliance. ZigBee. Specification, Connectivity Standards Alliance, 2003. `https://csa-iot.org`.

[28] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, et al. On scaling decentralized blockchains. In *International Conference on Financial Cryptography and Data Security*, pages 106–125. Springer, 2016.

[29] Elizabeth M Daly and Mads Haahr. Social network analysis for information flow in disconnected delay-tolerant MANETs. *IEEE Transactions on Mobile Computing*, 8(5):606–621, 2009.

[30] Francesco M Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. Reasoning in description logics. *Principles of Knowledge representation*, 1:191–236, 1996.

[31] Martin Duerst and Michel Suignard. Internationalized Resource Identifiers. RFC 3987, Internet Engineering Task Force, January 2005. `https://rfc-editor.org/rfc/rfc3987.txt`.

[32] Phan Minh Dung. On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and N-Person Games. *Artificial intelligence*, 77(2):321–357, 1995.

[33] Paul E. Dunne, Anthony Hunter, Peter McBurney, Simon Parsons, and Michael Wooldridge. Weighted argument systems: Basic definitions, algorithms, and complexity results. *Artificial Intelligence*, 175(2):457 – 486, 2011.

[34] Matthew English, Sören Auer, and John Domingue. Block Chain Technologies & The Semantic Web: A Framework for Symbiotic Development. In *Computer Science Conference for University of Bonn Students*, pages 47–61, 2016.

[35] Mauro Fadda, Matteo Anedda, Roberto Girau, Giovanni Pau, and Daniele D. Giusto. A social internet of things smart city solution for traffic and pollution monitoring in cagliari. *IEEE Internet of Things Journal*, 10(3):2373–2390, 2023.

[36] Ivan Farris, Roberto Girau, Michele Nitti, Luigi Atzori, Roberto Bruschi, Antonio Iera, and Giacomo Morabito. Taking the SIoT down from the cloud: Integrating the Social Internet of Things in the INPUT architecture. In *Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on*, pages 35–39. IEEE, 2015.

[37] I. Fette and A. Melnikov. The WebSocket Protocol. RFC 6455, IETF, December 2011. `https://tools.ietf.org/html/rfc6455`.

[38] Roberto Girau, Salvatore Martis, and Luigi Atzori. Lysis: a platform for IoT distributed applications over socially connected objects. *IEEE Internet of Things Journal*, 4(1):40–51, 2017.

[39] Nancy Gulati and Pankaj Deep Kaur. A game theoretic approach for conflict resolution in argumentation enabled social IoT networks. *Ad Hoc Networks*, 107:102222, 2020.

[40] Xiaofeng He, Yuchao Zhang, and Xiaotian Wang. A scalable nested blockchain framework with dynamic node selection approach for iot. In *2022 IEEE International Performance, Computing, and Communications Conference (IPCCC)*, pages 108–113, 2022.

[41] Matthew Horridge and Peter Patel-Schneider. OWL 2 Web Ontology Language Manchester Syntax (Second Edition). W3C note, W3C, December 2012. http://www.w3.org/TR/owl2-manchester-syntax/.

[42] Jiejun Hu, Martin Reed, Nikolaos Thomos, Mays F. Al-Naday, and Kun Yang. A dynamic service trading in a dlt-assisted industrial iot marketplace. *IEEE Transactions on Network and Service Management*, pages 1–1, 2022.

[43] Richard Hull, Vishal S Batra, Yi-Min Chen, Alin Deutsch, Fenno F Terry Heath III, and Victor Vianu. Towards a shared ledger business collaboration language based on data-aware processes. In *In-*

ternational Conference on Service-Oriented Computing, pages 18–36. Springer, 2016.

[44] Dina Hussein, Son N Han, Gyu Myoung Lee, and Noel Crespi. Social cloud-based cognitive reasoning for task-oriented recommendation. *IEEE Cloud Computing*, 2(6):10–19, 2015.

[45] Dina Hussein, Soochang Park, Son N Han, and Noel Crespi. Dynamic social structure of things: a contextual approach in CPSS. *IEEE Internet Computing*, 19(3):12–20, 2015.

[46] Florian Idelberger, Guido Governatori, Régis Riveret, and Giovanni Sartor. Evaluation of logic-based smart contracts for blockchain systems. In *International Symposium on Rules and Rule Markup Languages for the Semantic Web*, pages 167–183. Springer, 2016.

[47] IEC PAS 63088. *Smart manufacturing – Reference architecture model industry 4.0 (RAMI4.0).* IEC – International Electrotechnical Commission, 2017.

[48] Abdul Rehman Javed, Waqas Ahmed, Sharnil Pandya, Praveen Kumar Reddy Maddikunta, Mamoun Alazab, and Thippa Reddy Gadekallu. A survey of explainable artificial intelligence for smart cities. *Electronics*, 12(4), 2023.

[49] Upul Jayasinghe, Hyun-Woo Lee, and Gyu Myoung Lee. A Computational Model to Evaluate Honesty in Social Internet of Things. In *Proceedings of the Symposium on Applied Computing*, pages 1830–1835. ACM, 2017.

[50] Yiming Jiang, Chenxu Wang, Yawei Wang, and Lang Gao. A Cross-Chain Solution to Integrating Multiple Blockchains for IoT Data Management. *Sensors*, 19(9), 2019.

[51] Luo Kan, Yu Wei, Amjad Hafiz Muhammad, Wang Siyuan, Ling Chao Gao, and Hu Kai. A multiple blockchains architecture on inter-blockchain communication. In *2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pages 139–145, 2018.

[52] Jiawen Kang, Rong Yu, Xumin Huang, Maoqiang Wu, Sabita Maharjan, Shengli Xie, and Yan Zhang. Blockchain for secure and efficient data sharing in vehicular edge computing and networks. *IEEE Internet of Things*, 2018.

[53] Panagiotis Kasnesis, Lazaros Toumanidis, Dimitris Kogias, Charalampos Z Patrikakis, and Iakovos S Venieris. ASSIST: An agent-based SIoT simulator. In *Internet of Things (WF-IoT), 2016 IEEE 3rd World Forum on*, pages 353–358. IEEE, 2016.

[54] Henry M Kim and Marek Laskowski. Toward an ontology-driven blockchain design for supply-chain provenance. *Intelligent Systems in Accounting, Finance and Management*, 25(1):18–27, 2018.

[55] KNX association. KNX. Open standard, KNX association, 2002. `https://knx.org`.

[56] Kari Korpela, Jukka Hallikas, and Tomi Dahlberg. Digital Supply Chain Transformation toward Blockchain Integration. In *Proc. of 50th Hawaii International Conference on System Sciences*, pages 4182–4191, 2017.

[57] SK Lakshmanaprabu, K Shankar, Ashish Khanna, Deepak Gupta, Joel JPC Rodrigues, Plácido R Pinheiro, and Victor Hugo C De Albuquerque. Effective Features to Classify Big Data Using Social Internet of Things. *IEEE Access*, 6:24196–24204, 2018.

[58] Ali Li, Xiaozhen Ye, and Huansheng Ning. Thing Relation Modeling in the Internet of Things. *IEEE Access*, 5:17117–17125, 2017.

[59] L. Li and I. Horrocks. A software framework for matchmaking based on semantic web technology. *Int. Jour. of Electronic Commerce*, 8(4):39–60, 2004.

[60] Lulu Li, Huan Qu, H. Wang, Junyu Wang, Bozhi Wang, Wei Wang, Jinfei Xu, and Zhihui Wang. A blockchain-based product traceability system with off-chain epcis and iot device authentication. *Sensors (Basel, Switzerland)*, 22, 2022.

[61] Zhiyuan Li, Rulong Chen, Lu Liu, and Geyong Min. Dynamic resource discovery based on preference and movement pattern similarity for large-scale social Internet of Things. *IEEE Internet of Things Journal*, 3(4):581–589, 2016.

[62] Chih-Hao Lin, Pin-Han Ho, and Hong-Chuan Lin. Framework for NFC-based intelligent agents: a context-awareness enabler for Social Internet of Things. *International Journal of Distributed Sensor Networks*, 10(2), 2014.

[63] Kuan-Yu Lin and Hsi-Peng Lu. Why people use social networking sites: An empirical study integrating network externalities and motivation theory. *Computers in human behavior*, 27(3):1152–1161, 2011.

[64] Marco Lippi, Marco Mamei, Stefano Mariani, and Franco Zambonelli. An argumentation-based perspective over the social IoT. *IEEE Internet of Things Journal*, 5(4):2537–2547, 2018.

[65] Hong Liu, Yan Zhang, and Tao Yang. Blockchain-Enabled Security in Electric Vehicles Cloud and Edge Computing. *IEEE Network*, 32(3):78–83, 2018.

[66] Mengting Liu, F Richard Yu, Yinglei Teng, Victor CM Leung, and Mei Song. Computation offloading and content caching in wireless blockchain networks with mobile edge computing. *IEEE Transactions on Vehicular Technology*, 67(11):11008–11021, 2018.

[67] Zhiyuan Liu, Yang Liu, Qiang Meng, and Qixiu Cheng. A tailored machine learning approach for urban transport network flow estimation. *Transportation Research Part C Emerging Technologies*, 108, 09 2019.

[68] Ellie Lovellette, Henry Hexmoor, and Kane Rodriguez. Automated argumentation for collaboration among cyber-physical system actors at the edge of the Internet of Things. *Internet of Things*, 5:84–96, 2019.

[69] Viviana Mascardi, Angela Locoro, and Paolo Rosso. Automatic ontology matching via upper ontologies: A systematic evaluation.

IEEE Transactions on knowledge and data engineering, 22(5):609–623, 2009.

[70] Juri Mattila, Timo Seppälä, and Jan Holmström. Product-centric Information Management: A Case Study of a Shared Platform with Blockchain Technology. In *Berkeley Roundtable on the International Economy*, 2016.

[71] Matthias Mettler. Blockchain technology in healthcare: The revolution starts here. In *2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom)*, pages 1–3. IEEE, 2016.

[72] Noman Nasir Minhas and Maida Naveed. Using internet of things application for energy-efficient and lightweight internet of drones networks. *IT Professional*, 25(4):21–28, 2023.

[73] Martín O. Moguillansky and Guillermo R. Simari. A Generalized Abstract Argumentation Framework for Inconsistency-tolerant Ontology Reasoning. *Expert Systems with Applications*, 64(C):141–168, December 2016.

[74] Mehdi Montakhabi, Shenja van der Graaf, Akash Madhusudan, Roozbeh Sarenche, and Mustafa A. Mustafa. Fostering energy transition in smart cities: Dlts for peer-to-peer electricity trading. In *2021 17th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 466–472, 2021.

[75] Boris Motik, Bijan Parsia, and Peter Patel-Schneider. OWL 2 Web Ontology Language XML Serialization (Second Edition). Recommendation, W3C, December 2012. `https://www.w3.org/TR/owl2-xml-serialization`.

[76] Roopa M.S., Santosh Pattar, Rajkumar Buyya, Venugopal K.R., S.S. Iyengar, and L.M. Patnaik. Social internet of things (siot): Foundations, thrust areas, systematic review and future directions. *Computer Communications*, 139:32–57, 2019.

[77] Dae-Hyeok Mun, Minh Le Dinh, and Young-Woo Kwon. An assessment of Internet of Things protocols for resource-constrained appli-

cations. In *Computer Software and Applications Conference*, pages 555–560. IEEE, 2016.

[78] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[79] Alfredo Nascita, Antonio Montieri, Giuseppe Aceto, Domenico Ciuonzo, Valerio Persico, and Antonio Pescapè. Xai meets mobile traffic classification: Understanding and improving multimodal deep learning architectures. *IEEE Transactions on Network and Service Management*, PP, 07 2021.

[80] B. Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43(2):235–249, 1990.

[81] Michele Nitti, Roberto Girau, and Luigi Atzori. Trustworthiness Management in the Social Internet of Things. *IEEE Transactions on knowledge and data engineering*, 26(5):1253–1266, 2014.

[82] Michele Nitti, Maurizio Murroni, Mauro Fadda, and Luigi Atzori. Exploiting Social Internet of Things Features in Cognitive Radio. *IEEE Access*, 4:9204–9212, 2016.

[83] Alex Norta. Creation of smart-contracting collaborations for decentralized autonomous organizations. In *International Conference on Business Informatics Research*, pages 3–17. Springer, 2015.

[84] Yustus Eko Oktian, Sang-Gon Lee, and Hoon Jae Lee. Hierarchical multi-blockchain architecture for scalable internet of things environment. *Electronics*, 9(6), 2020.

[85] Kelly Olson, Mic Bowman, James Mitchell, Shawn Amundson, Dan Middleton, and Cian Montgomery. Sawtooth: An Introduction. Technical report, The Linux Foundation, 01 2018.

[86] Alfonso Panarello, Nachiket Tapas, Giovanni Merlino, Francesco Longo, and Antonio Puliafito. Blockchain and IoT Integration: A Systematic Survey. *Sensors*, 18(8):2575, 2018.

[87] Mrutyunjaya Panda and Ajith Abraham. Development of a reliable trust management model in social internet of things. *International*

Journal of Trust Management in Computing and Communications, 2(3):229–258, 2014.

[88] Bijan Parsia, Boris Motik, and Peter Patel-Schneider. OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition). W3C recommendation, W3C, December 2012. http://www.w3.org/TR/owl2-syntax/.

[89] Bijan Parsia, Sebastian Rudolph, Markus Krötzsch, Peter Patel-Schneider, and Pascal Hitzler. OWL 2 Web Ontology Language Primer (Second Edition). W3C Recommendation, W3C, December 2012. `http://www.w3.org/TR/owl2-primer`.

[90] Pankesh Patel, Muhammad Intizar Ali, and Amit Sheth. From raw data to smart manufacturing: Ai and semantic web of things for industry 4.0. *IEEE Intelligent Systems*, 33(4):79–86, 2018.

[91] Anand Paul, Awais Ahmad, M Mazhar Rathore, and Sohail Jabbar. Smartbuddy: Defining Human Nehaviors Using Big Data Analytics in Social Internet of Things. *IEEE Wireless Communications*, 23(5):68–74, 2016.

[92] Huma Pervez, Muhammad Muneeb, Muhammad Irfan, and Irfan Haq. A Comparative Analysis of DAG-Based Blockchain Architectures. pages 27–34, 12 2018.

[93] Antonio Pintus, Davide Carboni, and Andrea Piras. Paraimpu: a Platform for a Social Web of Things. In *Proceedings of the 21st International Conference on World Wide Web*, pages 401–404. ACM, 2012.

[94] Serguei Yu. Popov. The tangle. 2015.

[95] Nihar Pradhan, Akhilendra Singh, Sahil Verma, Kavita ., Marcin Wozniak, Jana Shafi, and Muhammad Fazal Ijaz. A blockchain based lightweight peer-to-peer energy trading framework for secured high throughput micro-transactions. *Scientific Reports*, (2022) 12:14523, 08 2022.

[96] Prud'hommeaux and editors E. Seaborne A. SPARQL Query Language for RDF, W3C Recommendation. http://www.w3.org/TR/rdf-sparql-query/, 2008.

[97] Eric Prud'hommeaux and Gavin Carothers. RDF 1.1 Turtle. Recommendation, W3C, February 2014. `https://www.w3.org/TR/turtle/`.

[98] Veena Pureswaran and Paul Brody. Device democracy: Saving the future of the Internet of Things. Technical report, IBM Institute for Business Value, September 2014. `http://www-935.ibm.com/services/us/gbs/thoughtleadership/internetofthings`.

[99] Haytham Qushtom, Jelena Mišić, Xiaolin Chang, and Vojislav Misic. A Scalable Two-Tier PBFT Consensus for Blockchain-Based IoT Data Recording. 01 2021.

[100] Azzurra Ragone, Michele Ruta, Eugenio Di Sciascio, and Francesco Donini. Bargaining agents in wireless contexts: An alternating-offers protocol for multi-issue bilateral negotiation in mobile marketplaces. volume 5692, pages 14–25, 09 2009.

[101] Borja Ramis Ferrer, Sergii Iarovyi, Luis Gonzalez, Andrei Lobov, and Jose L Martinez Lastra. Management of distributed knowledge encapsulated in embedded devices. *International Journal of Production Research*, pages 1–18, 2015.

[102] Iqra Rao, Miss Laiha Mat Kiah, M. Hameed, and Zain Memon. Scalability of blockchain: a comprehensive review and future research direction. *Cluster Computing*, pages 1–24, 02 2024.

[103] Shini Renjith, A. Sreekumar, and M. Jathavedan. An extensive study on the evolution of context-aware personalized travel recommender systems. *Information Processing & Management*, 57:102078, 07 2019.

[104] Ana Reyna, Cristian Martín, Jaime Chen, Enrique Soler, and Manuel Díaz. On blockchain and its integration with IoT. Challenges and opportunities. *Future Generation Computer Systems*, 88:179–190, 2018.

[105] Michele Ruta, Eugenio Di Sciascio, and Floriano Scioscia. Concept Abduction and Contraction in Semantic-based P2P Environments. *Web Intelligence and Agent Systems*, 9(3):179–207, 2011.

[106] Michele Ruta, Floriano Scioscia, Ivano Bilenchi, Filippo Gramegna, Giuseppe Loseto, Saverio Ieva, and Agnese Pinto. A multiplatform reasoning engine for the Semantic Web of Everything. *Journal of Web Semantics*, 73:100709, 2022.

[107] Michele Ruta, Floriano Scioscia, and Eugenio Di Sciascio. Enabling the semantic web of things: framework and architecture. *Sixth IEEE International Conference on Semantic Computing (ICSC 2012)*, pages 345–347, sep 2012. doi: 10.1109/ICSC.2012.42.

[108] Michele Ruta, Floriano Scioscia, Eugenio Di Sciascio, and Domenico Rotondi. Ubiquitous Knowledge Bases for the Semantic Web of Things. In *First Internet of Things International Forum*, November 2011.

[109] Michele Ruta, Floriano Scioscia, Filippo Gramegna, Saverio Ieva, Eugenio Di Sciascio, and Raffaello Perez De Vera. A Knowledge Fusion Approach for Context Awareness in Vehicular Networks. *IEEE Internet of Things Journal*, 5(4):2407–2419, 2018.

[110] Michele Ruta, Floriano Scioscia, Saverio Ieva, Giovanna Capurso, and Eugenio Di Sciascio. Semantic blockchain to improve scalability in the Internet of Things. *Open Journal of Internet Of Things*, 3(1):46–61, 2017.

[111] Michele Ruta, Floriano Scioscia, Saverio Ieva, Giuseppe Loseto, Agnese Pinto, and Arnaldo Tomasino. Blockchain and Knowledge Representation for Service-oriented Smart Mobility Platforms. *Blockchain: Research and Applications*, in press.

[112] Michele Ruta, Floriano Scioscia, Giuseppe Loseto, and Eugenio Di Sciascio. A semantic-enabled social network of devices for building automation. *IEEE Transactions on Industrial Informatics*, 13(6):3379–3388, Dec 2017.

[113] Michele Ruta, Floriano Scioscia, Giuseppe Loseto, Filippo Gramegna, Saverio Ieva, Agnese Pinto, and Eugenio Di Sciascio. Social Internet of Things for domotics: A knowledge-based approach over LDP-CoAP. *Semantic Web*, 9(6):781–802, 2018.

[114] Michele Ruta, Floriano Scioscia, Giuseppe Loseto, Agnese Pinto, and Eugenio Di Sciascio. Machine learning in the Internet of Things: A semantic-enhanced approach. *Semantic Web*, 10(1):183–204, 2019.

[115] Michele Ruta, Floriano Scioscia, Giuseppe Loseto, Agnese Pinto, Corrado Fasciano, Giovanna Capurso, and Eugenio Di Sciascio. Internet of Conscious Things: Ontology-Based Social Capabilities for Smart Objects. *Future Internet*, 16(9):327, 2024.

[116] Markus Sabadello, Manu Sporny, Drummond Reed, and Amy Guy. Decentralized identifiers (DIDs) v1.0. W3C recommendation, W3C, July 2022. https://www.w3.org/TR/2022/REC-did-core-20220719/.

[117] Sandeep Saharan, Seema Bawa, and Neeraj Kumar. Dynamic pricing techniques for intelligent transportation system in smart cities: A systematic review. *Computer Communications*, 150, 12 2019.

[118] Mohammad Samadi. Behavior-based decision making: A tutorial. *Dynamics and Control*, 6:1816–1840, 12 2018.

[119] Abdurrashid Ibrahim Sanka and Ray C.C. Cheung. A systematic review of blockchain scalability: Issues, solutions, analysis and future research. *Journal of Network and Computer Applications*, 195:103232, 2021.

[120] Manfred Schmidt-Schauß and Gert Smolka. Attributive concept descriptions with complements. *Artificial intelligence*, 48(1):1–26, 1991.

[121] Guus Schreiber and Fabien Gandon. RDF 1.1 XML syntax. W3C recommendation, W3C, February 2014. `http://www.w3.org/TR/rdf-syntax-grammar/`.

[122] Floriano Scioscia, Michele Ruta, Giuseppe Loseto, Filippo Gramegna, Saverio Ieva, Agnese Pinto, and Eugenio Di Sciascio. Mini-ME matchmaker and reasoner for the Semantic Web of Things. In *Innovations,*

Developments, and Applications of Semantic Web and Information Systems, pages 262–294. IGI Global, Hershey, PA, USA, 2018.

[123] Nigel Shadbolt, Tim Berners-Lee, and Wendy Hall. The Semantic Web revisited. *IEEE intelligent systems*, 21(3):96–101, 2006.

[124] Saima Shahab, Parul Agarwal, Tabish Mufti, and Ahmed J Obaid. SIoT (Social Internet of Things): a review. *ICT Analysis and Applications*, pages 289–297, 2022.

[125] Ayesha Siddiqa, Munam Ali Shah, Hasan Ali Khattak, Adnan Akhunzada, Ihsan Ali, Zaidi Bin Razak, and Abdullah Gani. Social Internet of Vehicles: Complexity, Adaptivity, Issues and Beyond. *IEEE Access*, 6, 2018.

[126] Steffen Staab and Rudi Studer. *Handbook on ontologies*. Springer Science & Business Media, 2010.

[127] Maria Rosaria Stufano Melone, Stefano Borgo, and Domenico Camarda. Digital twins of cities vs. digital twins for cities. In Alessandro Marucci, Francesco Zullo, Lorena Fiorini, and Lucia Saganeiti, editors, *Innovation in Urban and Regional Planning*, pages 192–203, Cham, 2024. Springer Nature Switzerland.

[128] Nick Szabo. Formalizing and securing relationships on public networks. *First Monday*, 2(9), 1997.

[129] Christoph Tempich, H Sofia Pinto, York Sure, and Steffen Staab. An argumentation ontology for DIstributed, Loosely-controlled and evolvInG Engineering processes of oNTologies (DILIGENT). In *European Semantic Web Conference*, pages 241–256. Springer, 2005.

[130] Nikolay Teslya and Alexander Smirnov. Blockchain-based framework for ontology-oriented robots' coalition formation in cyberphysical systems. In *MATEC Web of Conferences*, volume 161, page 03018. EDP Sciences, 2018.

[131] Florian Toqué, Mostepha Khouadjia, Etienne Côme, Martin Trépanier, and Latifa Oukhellou. Short & long term forecasting of multimodal transport passenger flows with machine learning methods. pages 560–566, 10 2017.

[132] Orfefs Voutyras, Panagiotis Bourelos, Spyridon Gogouvitis, Dimosthenis Kyriazis, and Theodora Varvarigou. Social monitoring and social analysis in Internet of Things virtual networks. In *Intelligence in Next Generation Networks (ICIN), 2015 18th International Conference on*, pages 244–251. IEEE, 2015.

[133] Marko Vukolić. The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication. In *International Workshop on Open Problems in Network Security*, pages 112–125. Springer, 2015.

[134] W3C OWL Working Group. OWL 2 Web Ontology Language Manchester Syntax (Second Edition). W3C Working Group Note, W3C, December 2012. https://www.w3.org/TR/owl2-manchester-syntax/.

[135] Huaiqing Wang, Kun Chen, and Dongming Xu. A maturity model for blockchain adoption. *Financial Innovation*, 2(12), 2016.

[136] Jin Wang, Jiayi Cao, Bin Li, Sungyoung Lee, and R Simon Sherratt. Bio-inspired ant colony optimization based clustering algorithm with mobile sinks for applications in consumer home automation networks. *IEEE Transactions on Consumer Electronics*, 61(4):438–444, 2015.

[137] K. Wang, H. Yin, W. Quan, and G. Min. Enabling Collaborative Edge Computing for Software Defined Vehicular Networks. *IEEE Network*, 32(5):112–117, 2018.

[138] Qin Wang, Xinqi Zhu, Yiyang Ni, Li Gu, and Hongbo Zhu. Blockchain for the IoT and industrial IoT: A review. *Internet of Things*, 10:100081, 2020.

[139] Tianyu Wang, Qian Wang, Zhaoyan Shen, Zhiping Jia, and Zili Shao. Understanding Characteristics and System Implications of DAG-Based Blockchain in IoT Environments. *IEEE Internet of Things Journal*, 9(16):14478–14489, 2022.

[140] Jianqing Wu, Luping Zhou, Chen Cai, Jun Shen, Sim Lau, and Jianming Yong. Data Fusion for MaaS: Opportunities and Challenges. In *IEEE 22nd International Conference on Computer Supported Cooperative Work in Design*, pages 642–647, 06 2018.

[141] Zheng Yan, Peng Zhang, and Athanasios V Vasilakos. A survey on trust management for Internet of Things. *Journal of network and computer applications*, 42:120–134, 2014.

[142] Roberto Yus and Primal Pappachan. Are Apps Going Semantic? A Systematic Review of Semantic Mobile Applications. In *1st International Workshop on Mobile Deployment of Semantic Technologies (MoDeST)*, volume 1506 of *CEUR Workshop Proceedings*, pages 2–13, Aachen, Germany, 2015. CEUR-WS.

[143] Long Zhang, Zhen Zhao, Qiwu Wu, Hui Zhao, Haitao Xu, and Xiaobo Wu. Energy-aware dynamic resource allocation in uav assisted mobile edge computing over social internet of vehicles. *IEEE Access*, 6:56700–56715, 2018.

[144] Qiheng Zhou, Huawei Huang, Zibin Zheng, and Jing Bian. Solutions to scalability of blockchain: A survey. *IEEE Access*, 8:16440–16455, 2020.

[145] Mirko Zichichi, Stefano Ferretti, and Víctor Rodríguez-Doncel. Decentralized Personal Data Marketplaces: How Participation in a DAO Can Support the Production of Citizen-Generated Data. *Sensors*, 22(16), 2022.

List of publications

International Journals

1. Michele Ruta, Floriano Scioscia, Saverio Ieva, Giuseppe Loseto, Agnese Pinto, Arnaldo Tomasino. Blockchain and Knowledge Representation for Service-oriented Smart Mobility Platforms, *Blockchain: Research and Applications* (SJR: Q1; IF: 6.9), Elsevier, *accepted, in press.*

Peer-Reviewed International Conferences

1. Agnese Pinto, Saverio Ieva, Arnaldo Tomasino, Giuseppe Loseto, Floriano Scioscia, Michele Ruta, Francesco De Feudis. A framework for automatic Knowledge Base generation from observation data sets, 2nd International Workshop on the Semantic WEb of EveryThing (SWEET 2023), 2023.

2. Giuseppe Loseto, Giuseppe Patella, Carmelo Ardito, Saverio Ieva, Arnaldo Tomasino, Lorenzo E. Malgieri, Michele Ruta. Towards a knowledge-based approach for digitalizing integrated care pathways, In International Workshop on Human-Centered Software Engineering (HCSE 2023), 2023.

3. Filippo Gramegna, Arnaldo Tomasino, Giuseppe Loseto, Floriano Scioscia, Michele Ruta. Features and capabilities of a blockchain-based ridesharing enhanced with semantics, In Current Trends in Web Engineering. ICWE 2022 International Workshops, BECS, SWEET and WALS.

Peer-Reviewed National Conferences

1. Floriano Scioscia, Giuseppe Loseto, Arnaldo Tomasino, Ivano Bilenchi, Filippo Gramegna, Saverio Ieva, Agnese Pinto, Eugenio Di Sciascio, Michele Ruta. Embedded reasoning for UAV operations: towards real-time efficiency and trustworthy autonomy, In 9th Italian conference on ICT for Smart Cities & Communities (I-CiTies 2023), 2023.

2. Filippo Gramegna, Arnaldo Tomasino, Saverio Ieva, Ivano Bilenchi, Agnese Pinto, Giuseppe Loseto, Floriano Scioscia, Michele Ruta. RideMATCHain: a Semantic-enhanced Blockchain Marketplace for Ridesharing, In 8th Italian Conference on ICT for Smart Cities And Communities (I-CiTies 2022), 2022.

3. Ivano Bilenchi, Arnaldo Tomasino, Filippo Gramegna, Saverio Ieva, Agnese Pinto, Giuseppe Loseto, Floriano Scioscia, Michele Ruta. Knowledge Representation and Reasoning for Unmanned Aerial Vehicle Intelligence, In 7th Italian Workshop on Embedded Systems (IWES 2022), 2022.