# Features and capabilities of a blockchain-based ridesharing enhanced with semantics

Filippo Gramegna[1 [0000−0003−4162−957X]], Arnaldo Tomasino[1 [0000−0002−6144−4647]], Giuseppe Loseto[2 [0000−0002−7995−8494]], Floriano Scioscia[1 [0000−0002−7859−9602]⋆], and Michele Ruta[1 [0000−0003−2125−327X]]

[1] Polytechnic University of Bari, Bari 70125, Italy
{name.surname}@poliba.it
[2] LUM University "Giuseppe Degennaro", Casamassima BA 70010, Italy
loseto@lum.it

**Abstract.** The integration of blockchain platforms with Artificial Intelligence (AI) technologies is increasingly frequent in research and industry. Semantic Web technologies provide peculiar opportunities, since they have been devised for information interoperability at Web scale and enable intelligent software agents for knowledge-driven task automation and decision support. The paper presents a blockchain platform extending Hyperledger Sawtooth with Knowledge Representation and Reasoning features. Semantic Smart Contracts allow annotated resource registration, discovery –leveraging non-standard inference services for semantic matchmaking–, explanation –exploiting logic-based justification provided by the adopted inferences– and selection. A prototypical semantic-enhanced ridesharing service has been built to validate the proposed framework and a time-series data collection infrastructure has been deployed for performance analysis.

**Keywords:** Knowledge Representation and Reasoning · Semantic matchmaking · Distributed Ledger Technologies · Time-series data analysis · Intelligent Transportation Systems

## 1 Introduction

In conventional distributed databases, a trusted intermediary is needed to ensure transaction integrity. This can be a strong limitation for several classes of applications, including Intelligent Transportation Systems (ITSs) and marketplaces open to the general public. To overcome this issue, *blockchain* has emerged as a family of data structures and protocols for peer-to-peer *trustless* distributed transactional systems. Blockchain platforms approve transactions by means of peer-to-peer *consensus* protocols. Transactions are usually grouped in *blocks*, which are appended sequentially. Alternative emerging proposals are based on Directed Acyclic Graphs (DAG), and an overall lively landscape of research on

⋆ Corresponding author Tel.: +39-080-596-3515; fax: +39-080-596-3410

*Distributed Ledger Technologies* (DLTs) is envisioning various kinds of architectural and algorithmic variations.

Many DLTs can host *Smart Contracts* (SCs), *i.e.*, programs encoding and enforcing terms of an agreement among two or more parties as executable procedures. This fosters the integration of DLTs with Artificial Intelligence (AI) technologies. In particular, Semantic Web technologies –grounded on formal Knowledge Representation and Reasoning– appear as a strong candidate for enhancing DLTs, since they have been devised to grant information interoperability at Web scale and to enable intelligent software agents for data-driven task automation or user decision support, a role which can be played by SCs in DLTs.

This paper proposes a semantic-enhanced blockchain platform for Service-Oriented Architectures (SOAs) and resource marketplaces. The proposal extends the *Hyperledger Sawtooth* project[3] [1] with semantic SCs for managing the registration, discovery, explanation and selection steps of service/resource lifecycle. Discovery exploits semantic matchmaking by means of non-standard inference services, capable of ranking a set of available resources by semantic affinity w.r.t. a request, where the request and the resources are annotated referring to a common domain ontology. The adopted inferences also grant logic-based explanation of outcomes, thus making requesters not only trust the the blockchain platform for transaction security, but also AI facilities by means of interpretable results.

In order to validate the correctness and usefulness of the proposal, a prototypical semantic-enhanced *carpooling* (a.k.a. *ridesharing*) service has been built. It allows detailed specification of vehicle and driver characteristics as well as passenger preferences, finding the best matches. For performance analysis of the proposed prototype, a time-series data analysis architecture has been deployed.

The remainder of the paper is as follows. Section 2 recalls the main aspects of Hyperledger Sawtooth. then Section 3 describes in detail the proposed framework. The ridesharing service is presented in Section 4, while Section 5 outlines the time-series data analysis infrastructure. A discussion of related work is in Section 6, before conclusion.

## 2 Background: Hyperledger Sawtooth

The proposed framework leverages the Hyperledger Sawtooth blockchain, an open source distributed ledger originally aimed for enterprise applications [1]. Sawtooth architecture provides a high decoupling among components, particularly suitable for IoT scenarios to enable transaction processing and data validation on heterogeneous devices. Moreover, the clear separation between application and functional system layers allows developers to write Smart Contracts (SCs) logic in multiple high-level programming languages. In Sawtooth, a client initially signs transactions and groups them into a *batch*, which is the atomic unit of change. If one of the transactions fails, the entire batch is aborted. Signed batches are then sent to a peer node, called *validator*, which is responsible for:

[3] Hyperledger Sawtooth: `https://www.hyperledger.org/use/sawtooth`

(i) processing transactions in parallel in order to improve performance, and (ii) publishing blocks of batches according to a consensus protocol [2]. Available consensus protocols are the Proof of Elapsed Time (PoET) –an extended version of the Practical Byzantine Fault Tolerance (PBFT) algorithm– Raft and Devmod, a simplified random-leader algorithm useful only for development and testing purposes [3]. Furthermore, the blockchain global state is stored and synchronized in a single instance of a *Merkle-Radix* tree on each validator, where each leaf address refers to serialized data related to a specific state change in a transaction.

## 3 Knowledge representation and blockchain

This section outlines the proposed DLT architecture and semantic-enhanced Smart Contracts.

### 3.1 Knowledge-based distributed ledgers

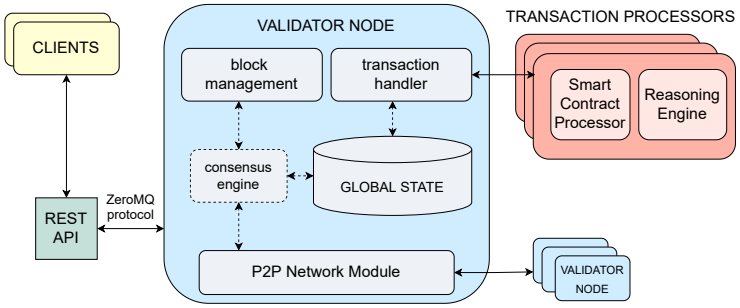


**Fig. 1.** Semantic-enhanced blockchain architecture

Figure 1 depicts the functional blocks of the Hyperledger Sawtooth architecture[4] enanched with semantic capabilities. The main components are:

- a set of **Transaction Processors** (TPs) executing transactions according to the rules associated with a *Transaction Family* (TF). A TF defines a transaction payload format, a model for storing information in the global state and a procedure for validating transactions and updating the state based on the payload;
- **Validator Nodes** which manage the global state and contains other subcomponents to be able to: (i) interact with TPs, (ii) coordinate the communication with their peers to apply the consensus procedure, (iii) verify incoming blocks and create new ones;

[4] Sawtooth documentation: `https://sawtooth.hyperledger.org/docs/1.2/`

– **Clients** which query Validator Nodes to known the state of the blockchain and submit batches. A REST API (REpresentational State Transfer Application Programming Interface) acts as a gateway between Clients and Validator Nodes, communicating through the *ZeroMQ* protocol[5]. Alternatively, Clients can directly talk to a Validator Node exchanging ZeroMQ messages.

The proposed framework extends the functionalities of basic TPs, providing a set of SCs that are able to manage assets annotated w.r.t. a domain ontology and perform semantic tasks. According to the modelled scenario, assets can represent physical or digital resources, as well as service instances. To orchestrate semantically annotated assets, a specific TF was defined with the implementation of the corresponding Smart Contract Processor. It interfaces with the *Tiny-ME* reasoning engine [9] to carry out the *Registration*, *Discovery*, *Explanation* and *Selection* semantic-enabled SCs. Using a blockchain substratum guarantees that outcomes are validated by consensus and data is always distributed consistently and synchronized among all participants in the network.

### 3.2 Semantic-enabled smart contracts

Semantic-enabled SCs are based on a Description Logics-based matchmaking framework [9], able not only to retrieve a set of resources compliant with a given request, but also to rank them according to a relevance score. In fact, in advanced scenarios full matches are rare and incompatibility often occurs when matching articulate descriptions. In order to achieve a granular ranking of potential and partial matches, non-standard inference services that enhance classic *Satisfiability* and *Subsumption* checks can be exploited. Given a request $R$ and a resource $S$ annotated w.r.t. a common ontology $\mathcal{T}$:

– if the *Satisfiability* check states that $R$ and $S$ are not compatible (*i.e.*, their conjunction is unsatisfiable), **Concept Contraction** detects: (i) what part $G$ (*Give up*) of $R$ clashes with $S$, (ii) a contracted version $K$ (*Keep*) of the original request such that it is compatible with $S$;
– if the *Subsumption* check determines that $S$ does not fully satisfy $R$ (*i.e.*, $S$ is not more specific than $R$), **Concept Abduction** identifies a concept $H$ (*Hypothesis*) representing what is missing in $S$ in order to reach a full match, in the Open World Assumption.

Furthermore, penalty functions based on the Conjunctive Normal Form for DL concept expressions can be associated to both $G$ and $H$, in order to compute a semantic distance of each available resource w.r.t. a given request. The introduced semantic-enabled SCs are detailed below:

– **Registration**. As depicted in Figure 2, to store a specific resource into the blockchain, a Client can send a registration transaction to a Validator Node specifying: (i) the resource Uniform Resource Identifier (URI); (ii) its

[5] ZeroMQ: `https://zeromq.org/`

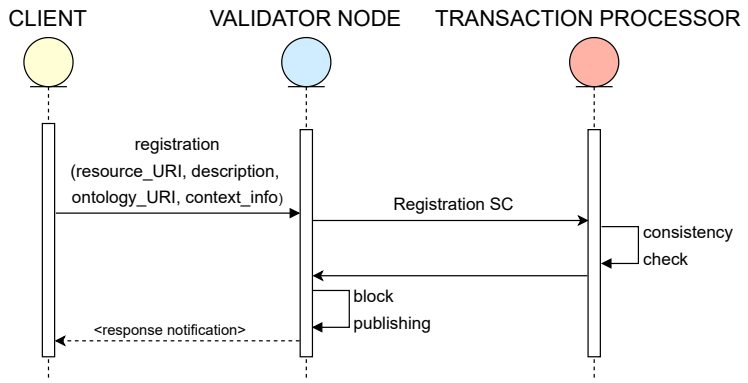


**Fig. 2.** Registration SC sequence diagram

semantic description, *i.e.*, a concept expression w.r.t. a reference ontology; (iii) the ontology URI; (iv) other context-related information (*e.g.*, resource price). The ontology URI is essential to uniquely identify resources, because multiple domains can coexist and associated ontologies can be stored in the same blockchain. Then the Validator invokes the `Registration SC`, which carries out a consistency check to verify the involved resource is satisfiable w.r.t. the referred ontology. If it fails, the transaction is aborted, otherwise a confirmation is sent back to the Client.

- **Resource Discovery**. In order to search for (a set of) items, as depicted in Figure 3, the requester sends to a Validator a discovery transaction detailing: (i) the URI of the reference ontology; (ii) the semantic description of the request, specifying desired features and constraints; (iii) other contextual information (*e.g.*, minimum semantic relevance threshold, maximum number of results to be returned or the price the requester is willing to pay). The `Discovery SC` is triggered by the Validator Node and a semantic matchmaking task [9], based on the **Concept Abduction** and **Concept Contraction** inference services, is carried out comparing the request annotation with the descriptions of all the available resources. A list of ranked resource URIs is returned to the Client.
- **Explanation**. This optional step allows a requester to get a justification of the matchmaking outcome and is useful to trigger a request refinement process. Referring to the sequence diagram in Figure 3, the submitted explanation transaction specifies the semantic description of the request and the URI of the discovered resource. The `Explanation SC` invoked by the Validator Node replies with the related matchmaking outcome explanation, structured as: (i) semantic affinity score in the [0,1] interval, computed by means of the semantic penalty and other domain-dependent contextual parameters, (*i.e.* physical distance, price); (ii) a description of the related reasoning outcome, in terms of $G$, $K$ or $H$ concept expressions.

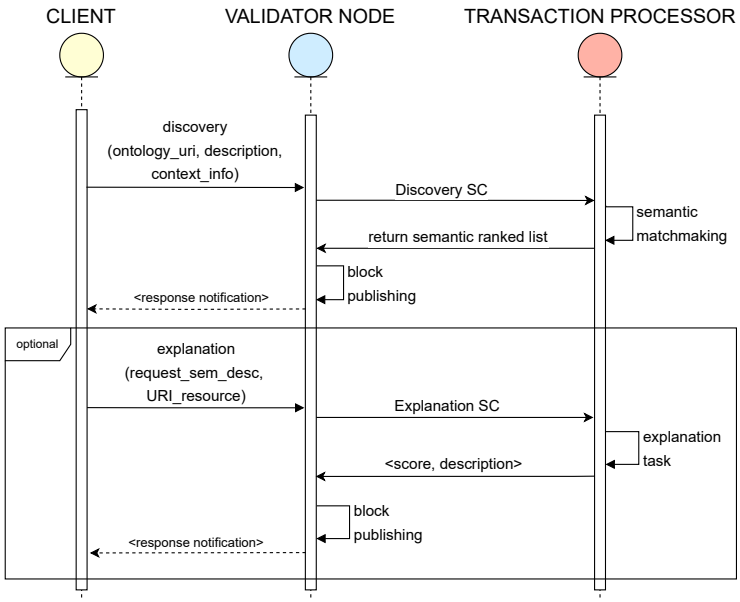


**Fig. 3.** Resource Discovery and Explanation SCs sequence diagram

– **Selection**. After receiving a set of results, the requester can choose the desired resource(s). The Client submits a selection transaction specifying the resource URI and optional contextual data. The `Selection SC` is thus invoked: it is responsible to update the selected resource description with additional compatible features resulting from the request as well as to reply with a properly usable resource representation (*e.g.*, an interface endpoint or a further SC to be invoked).

Regardless of the matchmaking result, each completed transaction is published on the blockchain, exploiting the consensus algorithm for robustness, traceability and accountability purposes.

## 4 RideMATCHain: an advanced ridesharing service

A practical example is described from a prototypical carpooling service named *RideMATCHain*[6], in order to better explain and underline the benefits of the proposed framework. 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 five drivers currently available for ridesharing through the service with their cars. The problem

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

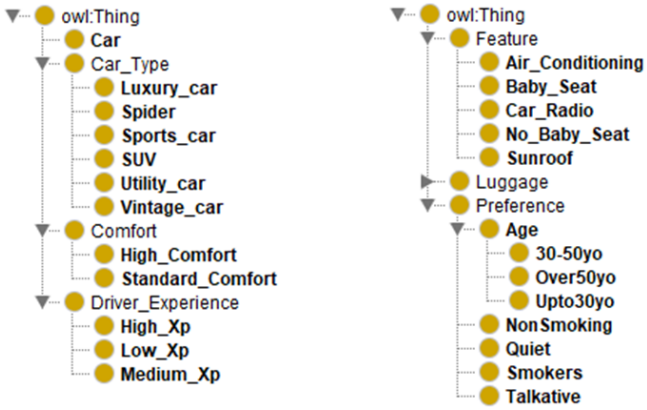


**Fig. 4.** Reference ontology top-level classes

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 4. Each available ride is a resource, represented an asset stored into the blockchain and endowed with a semantic annotation. The annotation is split into two parts: the first one is constant over time and summarizes both the set of vehicle features relevant to the service and the driver's profile; the second part specifies information about the space available for passengers and their luggage. An excerpt of the annotations linked to each car is shown below in Manchester syntax [4]:

```
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)
```

Previous 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 3.2. 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 seating positions - one of them equipped with baby seat - and enough space for a medium-size luggage and a stroller. 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 detailed in [9]. 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 are 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. An explanation for this can be obtained 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 to renew the blockchain internal status and repeat the allocation task for the remaining passengers, verifying any incompatibility constraints.

## 5 Time series-based performance analysis

In order to measure hardware and operational performances of the implemented framework, time-series data analysis tools have been integrated in the overall system architecture. Such an approach is useful to monitor and detect potential faults within a complex distributed system such as a blockchain.

### 5.1 Involved components and their deployment

In order to collect time-series metrics, the following new components have been deployed in the architecture described in Section 3.1:

- **InfluxDB**[7], an open-source time-series database that acts as collector for timestamped data. It stores time related data in a set of *measurements*, comparable to tables of a relational database. A measurement's record, a.k.a. *point*, refers to a specific observation and has a timestamp in addition to values organized in columns. InfluxDB has been chosen due to its high compatibility with both the *Telegraf* metrics collection tool and the *Grafana* dashboarding system.
- **Telegraf**[8], a server agent able to collect a set of predefined metrics from different data sources (databases, systems or IoT sensors) and forward them to multiple types of Data Base Management Systems. A configuration file contains details regarding *InfluxDB* connection, database and operations. The main advantage is that it has a little memory footprint and does not affect performances of the system where it is executed.
- **Grafana**[9], a Web-based open-source visualization tool useful to get a comprehensive view of all the collected parameters through a set of charts. Graphs show data according to predefined queries on a specific data source and can be organized in different dashboards.

Figure 5 clarifies the components interconnection. InfluxDB and Grafana are deployed as separate *Docker*[10] containers, while Telegraf runs as a background process within a Validator Node container, which also provides the required configuration file. All application services have been defined and orchestrated through *Docker Compose*[11].

### 5.2 Computed metrics and data visualization

The data collection process involves three different groups of parameters:

[7] InfluxDB: `https://www.influxdata.com/`
[8] Telegraf: `https://www.influxdata.com/time-series-platform/telegraf/`
[9] Grafana: `https://grafana.com/`
[10] Docker: `https://www.docker.com/`
[11] Docker Compose: `https://docs.docker.com/compose/`

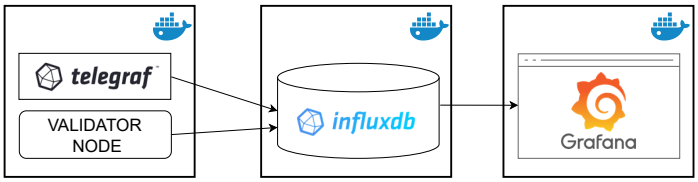


**Fig. 5.** Deployment of metrics collection components

- measures related to the hardware where the *Telegraf* process is running. Referring to the diagram depicted in Figure 5, they include the performance of the container where the Validator Node process runs: CPU load, available memory, hard disk activities (*i.e.*, percentage of waiting time for I/O tasks, disk read/write speed); operating system information (*i.e.*, the number of context switches that the system underwent);
- metrics directly retrieved by the Validator Node process, such as: number of blocks; pending, committed and published transactions in a time interval; data rates of exchanged messages and response times. While starting a Validator Node process, a set of parameters can be specified in order to collect and send proper data to the InfluxDB instance.
- performance related to the semantic tasks described in Section 3.2. They are directly collected by the implemented semantic-enabled SCs and measure the average time taken to complete a single semantic task of registration, discovery, explanation and selection.

As shown in Figure 6, metrics can be organized in different *Grafana* dashboards and are useful for trend analysis regarding the operations of the blockchain with respect to the user needs and behaviors.

## 6 Related work

In order to comply with requirements and needs in ITS, blockchains are leveraged to enable novel transparent and trustless interaction models [8]. Previous modeling approaches have been extended to combine IoT with blockchain technologies for different ITS scenarios [14] [5]. The main goal is to explore and improve data exchange, interoperability and cooperation among the various actors. SCs prove particularly useful in this perspective, as they help managing the mapping of new resources and a reliable storage of their (often complex) data [11] [13].

In order to further improve information representation and sharing among heterogeneous hosts leveraging SCs, the approach developed in [10] has exploited Semantic Web technologies to provide blockchain assets with a semantic-based structured representation. The advantages of trustworthiness and traceability offered by a blockchain infrastructure have also been enhanced with semantic-based services to discover and negotiate available resources. In Liu *et al.* [6] decentralized energy and charging service marketplaces for electric vehicles and

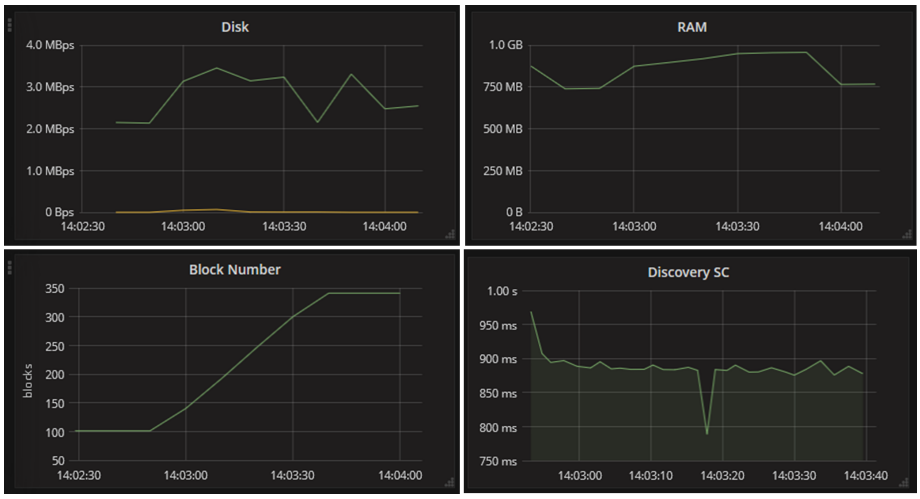


**Fig. 6.** Dashboards for system and blockchain related metrics

Smart Grid integration are presented as the most suitable scenarios for such services. Despite the benefits blockchain could offer in the automotive domain, many are the open issues related to vehicular data.

Standardization of information models is required for interoperability and data sharing, but it can be hindered by the historically competitive and secretive nature of the automotive sector. Initiatives have been recently launched to take steps ahead against this limitation. A hierarchically organized information model about vehicles data, as well as an architecture for a cloud-based Big Data marketplace, has been proposed in the *AutoMat* [7] Horizon 2020 project, albeit still not relying on a blockchain substratum. Conversely, the BMW Group *VerifyCar* project, based on the *VeChainThor* [12] blockchain, has introduced an experimental platform where up-to-date information about mileage, accident history, inspections, maintenance procedures and other useful information related to the lifecycle of a vehicle are stored and can be properly accessed by authorized parties at any time and cannot be tampered with.

## 7 Conclusion

The paper has introduced a semantic-enhanced DLT platform, based on the Hyperledger Sawtooth blockchain. Semantic Smart Contracts have been integrated to manage the fundamental steps of the lifecycle of semantically annotated resource descriptions: registration, discovery and selection. The adoption of non-standard inference services for semantic matchmaking non only provides a fine-grained ranking of resources w.r.t. a request, but also enables an explanation SC to make discovery outcomes fully interpretable. A prototypical ridesharing service has been built on top of the proposed platform in order to validate the

correctness and usefulness of the proposal, while a performance analysis architecture has been defined and deployed.

Future work includes three main directions: exploit the devised time-series data collection infrastructure to carry out thorough experimental performance analysis, even involving resource-constrained IoT nodes, in order to validate the computational feasibility and scalability of the proposal; expand the ridesharing service prototype toward a viable commercial proposition; extend the blockchain architecture with further semantic SCs to provide more advanced resource management capabilities, including resource composition, substitution, and negotiation.

## References

1. Aggarwal, S., Kumar, N.: Hyperledger. In: Advances in Computers, vol. 121, pp. 323–343. Elsevier (2021)
2. Ampel, B., Patton, M., Chen, H.: Performance Modeling of Hyperledger Sawtooth Blockchain. In: 2019 IEEE International Conference on Intelligence and Security Informatics (ISI). pp. 59–61 (2019)
3. Cachin, C., Vukolić, M.: Blockchain Consensus Protocols in the Wild. In: 31 International Symposium on Distributed Computing. pp. 19–34 (2017)
4. Horridge, M., Patel-Schneider, P.: OWL 2 Web Ontology Language Manchester Syntax (Second Edition). W3C note, W3C (Dec 2012), http://www.w3.org/TR/owl2-manchester-syntax/
5. Humayun, M., Jhanjhi, N., Hamid, B., Ahmed, G.: Emerging smart logistics and transportation using IoT and blockchain. IEEE Internet of Things Magazine **3**(2), 58–62 (2020)
6. Liu, H., Zhang, Y., Yang, T.: Blockchain-Enabled Security in Electric Vehicles Cloud and Edge Computing. IEEE Network **32**(3), 78–83 (2018)
7. Pillmann, J., Wietfeld, C., Zarcula, A., Raugust, T., Alonso, D.C.: Novel common vehicle information model (CVIM) for future automotive vehicle big data marketplaces. In: Intelligent Vehicles Symposium (IV). pp. 1910–1915. IEEE (2017)
8. Pureswaran, V., Brody, P.: Device democracy: Saving the future of the Internet of Things. Tech. rep., IBM Institute for Business Value (Sep 2014)
9. Ruta, M., Scioscia, F., Bilenchi, I., Gramegna, F., Loseto, G., Ieva, S., Pinto, A.: A multiplatform reasoning engine for the semantic web of everything. Journal of Web Semantics **73**, 100709 (2022)
10. Ruta, M., Scioscia, F., Ieva, S., Capurso, G., Di Sciascio, E.: Semantic blockchain to improve scalability in the Internet of Things. Open Journal of Internet Of Things (OJIOT) **3**(1), 46–61 (2017)
11. Valtanen, K., Backman, J., Yrjölä, S.: Creating value through blockchain powered resource configurations: Analysis of 5G network slice brokering case. In: 2018 IEEE Wireless Communications and Networking Conference Workshops. pp. 185–190. IEEE (2018)
12. VeChain Foundation: VeChain Whitepaper 2.0 – Creating valuable TXs (Dec 2019)
13. Xu, W., et al.: Internet of Vehicles in Big Data Era. IEEE/CAA Journal of Automatica Sinica **5**(1), 19–35 (2017)
14. Yuan, Y., Wang, F.Y.: Towards blockchain-based intelligent transportation systems. In: 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC). pp. 2663–2668. IEEE (2016)