© 2017 by the authors; licensee RonPub, Lübeck, Germany. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution license (http://creativecommons.org/licenses/by/4.0/).

Open Access

Open Journal of Internet of Things (OJIOT)
Volume 3, Issue 1, 2017
http://www.ronpub.com/ojiot
ISSN 2364-7108

Semantic Blockchain to Improve Scalability in the Internet of Things

Michele Ruta, Floriano Scioscia, Saverio Ieva, Giovanna Capurso, Eugenio Di Sciascio

SisInfLab (Information Systems Laboratory), Department of Electrical and Information engineering,
Polytechnic University of Bari, Via Edoardo Orabona, 4, I-70125 Bari, Italy,
{michele.ruta,floriano.scioscia,saverio.ieva,giovanna.capurso,eugenio.disciascio}@poliba.it

Abstract

Generally scarce computational and memory resource availability is a well known problem for the IoT, whose intrinsic volatility makes complex applications unfeasible. Noteworthy efforts in overcoming unpredictability (particularly in case of large dimensions) are the ones integrating Knowledge Representation technologies to build the so-called Semantic Web of Things (SWoT). In spite of allowed advanced discovery features, transactions in the SWoT still suffer from not viable trust management strategies. Given its intrinsic characteristics, blockchain technology appears as interesting from this perspective: a semantic resource/service discovery layer built upon a basic blockchain infrastructure gains a consensus validation. This paper proposes a novel Service-Oriented Architecture (SOA) based on a semantic blockchain for registration, discovery, selection and payment. Such operations are implemented as smart contracts, allowing distributed execution and trust. Reported experiments early assess the sustainability of the proposal.

Type of Paper and Keywords

Regular research paper: *Blockchain, Service-Oriented Architecture, Semantic Web of Things, Semantic Web, Internet of Things, Matchmaking, Ubiquitous Computing*

1 Introduction

Talking about *blockchain* technology, the most popular evidence is taken by *Bitcoin* currency[1], which is also the most controversial implementation of it, due to a billionaire worldwide hidden market of anonymous transactions outside the official regulatory control. However, in spite of the Bitcoin reputation, blockchain is an intrinsically positive technology with large affordability witnessed by several years of flawless adoption in disparate fields. It is grounded on a basic and fundamental concept: The trust of transactions in the digital world is strictly related to the confidence on a given "authority". It has to be noticed that today the assets made in a virtual way are more and more increasing. This is true not only for big companies, but also for private citizens relying on the digital world even more relevant elements of their daily life (personal information, photos, preferences profile, working career) a part from not negligible endowment amount.

The fact that certification authorities could be hacked and counterfeit poses serious security and privacy issues to the diffusion of any dematerialized transaction. This

[1] https://bitcoin.org/

This paper is accepted at the *International Workshop on Very Large Internet of Things (VLIoT 2017)* in conjunction with the VLDB 2017 Conference in Munich, Germany. The proceedings of VLIoT@VLDB 2017 are published in the Open Journal of Internet of Things (OJIOT) as special issue.

is the reason why blockchain could prove to be helpful. A blockchain is basically a database distributed and shared among several parties which records possible transactions happened in a given time span. The reliability of such a structure comes from the fact that every transaction is trusted by consensus of the majority of entities acting in the system through the execution of *smart contracts* [3], *i.e.*, software stubs which automatically process locally the terms of a contract. When a pre-configured condition in a smart contract is verified then general actions related to the agreement (*e.g.*, payments) are automatically performed.

As thought, blockchain technology is interesting from the Internet of Things (IoT) perspective: IoT suffers from the unpredictability of nodes inherited from the volatility of actors and appliances, which makes trust management difficult. Hence, scalability is kept low and possible applications are limited. Most important efforts in overcoming unpredictability focus on re-designing resource discovery approaches. Noteworthy is the integration of Knowledge Representation theory – and particularly Semantic Web technologies– at resource discovery level in IoT stacks. This effort starts off the so-called Semantic Web of Things (SWoT) [19]. It enables semantic-enhanced pervasive computing by embedding intelligence in both objects and ambient through the dissemination of a large number of micro-devices, each conveying a small fragment of semantically rich information.

SWoT opens the way toward an ontology-based resource/service discovery leveraging semantics of requests and resource descriptions to refine retrieval strategies. Unfortunately, such an approach alone leaves unsolved the problem of affordability of transactions after discovery has happened, which restrains a large exploitation of SWoT solutions in multiple contexts. Particularly, scalability is the most problematic aspect to be faced on when it comes to applications and scenarios involving large or very large IoT infrastructures. From this standpoint, blockchain technology could incorporate SWoT approaches providing interesting potentialities. A SWoT blockchain basically amounts to a Service-Oriented Architecture (SOA) for regulating registration, discovery, selection and –in case– payment operations. These subsequent tasks are intended as distributed smart contracts validated by consensus.

This paper proposes a possible framework for scalable large-dimensions SWoT systems. A semantic-based resource discovery layer is integrated in a basic blockchain infrastructure in a way that the blockchain adds verifiable records for every single transaction. A distinguishing feature of the systems is *logic-based explanation* of discovery outcomes, grounded on non-standard inference services for semantic matchmaking. The proposed system preserves fundamental blockchain features, also when size increases substantially. Particularly, the effective and secure structure of the chain is capable of detecting erroneous or malicious changes on a transaction block also in case of large amounts of volatile nodes. The following algorithm types are used to achieve consensus: (i) *Proof-of-Work* (PoW), based on solving a cryptographic problem in order to sign and validate a block of transactions; (ii) *Byzantine Fault-Tolerance* (BFT), a state-machine replication protocol which tolerates that among the N validating peers at most $f < N/3$ could be malicious. In order to incorporate a semantic-based resource discovery within the chain operations, annotations are registered as assets on the blockchain and nonstandard inference services in [20] are implemented as smart contracts, which are executed by validating peers.

A case study in the field of distributed and dynamic scheduling of production tasks for pharmaceutical factories is presented. It has the aim to clarify the proposal evidencing its benefits. In addition, the proposed framework has been implemented and tested on the *Hyperledger Iroha*[2] platform which adopts a BFT consensus algorithm named *Sumeragi*. It has been encapsulated in a *Docker*[3] container to simulate a large IoT infrastructure. Early experiments on the framework have been carried out and basically assess the feasibility of the approach.

The remainder of the paper is organized as follows. Next section highlights relevant background for the approach. A functional and architectural description of the proposed framework is given in Section 3 and Section 4 respectively. Section 5 proposes an exemplified case study, followed by Section 6, where experimental evaluation results are provided. Finally, Section 7 surveys most interesting related work before closing remarks.

2 BACKGROUND

To make the paper clear and self-contained, some details about blockchain technology and Semantic Web of Things are provided in this section.

2.1 Blockchain Basics

Blockchain is a data structure and protocol for *trustless* distributed transactional systems. In traditional distributed databases, a trusted intermediary is needed to guarantee irreversibility (*i.e.*, no committed transaction

[2] https://www.hyperledger.org/projects/iroha
[3] https://www.docker.com/

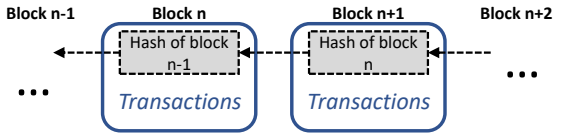

Figure 1: Blockchain structure

can be reverted or altered) and preventing censorship (*i.e.*, all valid transactions are committed). Blockchain systems avoid intermediaries by approving transactions through a distributed *consensus* protocol, which guarantees no single node – or small group of colluding nodes[4] – can force the addition, removal or modification of data. Transactions approved in a given time period – again, the window size depends on the particular blockchain system– are grouped in *blocks*. As depicted in Figure 1, for each block a hash is appended and computed not only on the contents of the block, but also on the hash of the previous block, thus forming a chain of blocks. This prevents tampering even with old blocks without consensus among the nodes.

Building on previous theoretical results on *Proof-of-Work* consensus algorithms [22], blockchain technology was born with *Bitcoin*, an open source platform for electronic currency. Bitcoin uses blockchain as a ledger to store currency transfer transactions. After the success of Bitcoin, many other blockchain-based electronic currency platforms have been introduced. At the same time, it has been realized that the underlying blockchain technology is an inherently general-purpose distributed database, enabling trustless collaboration of *Decentralized Autonomous Organizations* (DAOs). This feature enables practical implementations of the *Smart Contract* (SC) idea [21], *i.e.*, programs encoding and enforcing cooperative processes among two or more parties.

Originally, SCs required a trusted mediator, restraining a large development of the approach. Indeed, consensus about SCs in a blockchain is reached through a parallel execution in the network, effectively making every SC-enabled blockchain a general-purpose application platform based on a distributed Virtual Machine (VM). Many proposals have emerged, including proprietary platforms (*Ethereum*[5] is perhaps the most popular) and standardization efforts, such as the *Hyperledger*[6] initiative guided by the Linux Foundation. SC-based blockchains are being experienced in several financial and industry sectors. Internet of Things (IoT) scenarios are expected to be among the main application areas in the near future, as discussed in Section 7.

Several types of blockchain systems exist, based on the following key design decisions:

Network access policy. A blockchain network is *permission-less* if any node can join –even anonymously– at any time, or *permissioned* if a white-list of admitted nodes exists and nodes are uniquely identified. This choice has a deep impact on the blockchain design: Permission-less chains usually have to reward participants for their computational effort, *e.g.*, Bitcoin allows nodes to generate (*mine*) and keep new currency for the validation of transaction blocks. Permissioned chains are instead adopted in more controlled collaboration contexts, where access itself is a reward, as it enables selling and buying services or resources.

Consensus algorithm. Permission-less systems require stricter consensus methods, such as Proof-of-Work, which guarantee data security unless a large portion of nodes is colluding to subvert the blockchain. Permissioned systems –where each node is identifiable and accountable– may relax consensus constraints in order to reduce the computational load, by selecting simpler algorithms; Byzantine Fault Tolerance (BFT) variants [22] are often adopted.

Transaction model. In blockchain systems, *assets* can be registered and exchanged. At any time, each node typically owns some assets in a certain quantity. In the *unspent transaction outputs* (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. In the *account-based* model, instead, every node has an account reporting all its assets, which is updated by transactions. The UTXO model is similar to a bank statement of account; it allows simpler reconstruction of current state from a transaction log and is typically adopted by e-currency systems. The account-based model is more general, but it can make transaction processing slower; nevertheless, it is the only practical choice for general-purpose SC-based blockchains.

Smart contract language. Blockchains can adopt any formalism for SC specification and execution, such as procedural (imperative) languages or logical (declarative) languages or automata [9]. Industry proposals mostly adopt computationally complete programming languages, either existing (*e.g.*, Java in the *Iroha* framework of Hyperledger, exploited in this work) or created for the purpose (*e.g.*, Ethereum's *Solidity*).

Table 1 summarizes blockchain system features in typical e-currency and IoT solutions. A wider discussion of blockchain technology is in [3].

[4] *Small* is usually characterized as a maximum percentage of all participating nodes; the actual value depends on the particular consensus mechanism.

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

[6] Hyperledger: `https://www.hyperledger.org/`

Table 1: Typical blockchain features for e-currency and IoT solutions

Feature	E-currency	IoT
Access policy	Permission-less	Permissioned
Consensus algorithm	Proof-of-Work	BFT-like
Transaction model	UTXO	Account-based
Smart contracts	No	Yes
Programmable	No	Yes
General purpose	No	Yes
Transaction latency	Lower	Higher

2.2 Fundamentals of Semantic Web of Things

In latest years, the *Semantic Web of Things* (SWoT) [19] vision is merging the Semantic Web and the Internet of Things. Its goal is embedding intelligence into ordinary objects and environments via semantic-enhanced pervasive computing. Large numbers of heterogeneous micro-devices, each conveying a small amount of information, can interact autonomously to provide high-level services to users, via decision support and task automation.

Addressed IoT scenarios require flexibility and interoperability in information representation, management, exchange and discovery. Agents running on mobile and embedded devices should be able to discover dynamically the best available resources according to their requirements and preferences, in order to support user's tasks in a context-aware way. The Semantic Web provides standard knowledge representation models and languages – formally grounded on Description Logics (DLs)– for interoperable information modeling, sharing and automated inference.

Description Logics are a family of logic languages for Knowledge Representation in a decidable fragment of First Order Logic [1]. Basic DL syntax elements are:

Concept (a.k.a. *class*) names, standing for sets of objects, *e.g.*, *medicine, shape, sweetening_agent*;

Role (a.k.a. *object property*) names, linking pairs of objects in different concepts, like *hasDosage, hasShape*;

Individuals (a.k.a. *instances*), special named elements belonging to concepts, *e.g.*, *Acetylsalicylic_Acid_Regular, Coated_Caplet*.

Logical *constructors* combine the above elements to form concept and role *expressions*. Each DL has a specific set of constructors. The *conjunction* of concepts, usually denoted as $\sqcap$, is available in all DLs; Some DLs also use *disjunction* $\sqcup$ and *complement* $\neg$. Roles can be combined with concepts by means of *existential role quantification* (*e.g.*, $medicine \sqcap \exists hasDosageForm.coated$, representing medications available in coated form) and *universal role quantification* (*e.g.*, $medicine \sqcap \forall hasActiveIngredient.Acetylsalicylic\_Acid$, which describes the set of medications whose only active ingredient is acetylsalicylic acid). Other constructs involve counting, as *number restrictions*: $coated \sqcap \leq 2\, hasExcipient$ denotes coated medicines with at most two excipients, and $medicine \sqcap \geq 3\, hasDosageForm$ describes available in at least three dosage forms. Concept expressions can be used in *inclusion* (a.k.a. subsumption) and *definition* (a.k.a. equivalence) axioms, which model knowledge elicited for a given domain. A set of such axioms is called *Terminological Box (TBox)*, a.k.a. *ontology*. A set of individual axioms (a.k.a. facts) constitutes an *Assertion Box (ABox)*. TBox and ABox together form a *Knowledge Base (KB)*.

The proposed blockchain-based resource discovery approach leverages *semantic matchmaking*, *i.e.*, the process allowing the retrieval and ranking of the most relevant resources for a given request, where both resources and requests are concept expressions w.r.t. a common ontology $\mathcal{T}$. Given a request R and a resource S, *subsumption* checks whether all features in R are included in S; *satisfiability* checks whether any constraint in R contradicts some specification in S. Unfortunately these classic inference services enable only a Boolean *full match or no match* approach. This is inadequate in advanced scenarios, because full matches are rare and incompatibility is frequent when dealing with detailed concept expressions. Complex reasoning and query answering problems such as semantic matchmaking require further specialized reasoning services.

Therefore, the proposal exploits the following non-standard inferences [20] to determine a semantic ranking of resources w.r.t. a request and a logic-based explanation of outcomes:

Concept Contraction: if request R and resource S are not compatible, Contraction determines which part of R is conflicting with S. If one retracts conflicting requirements in R, G (for *Give up*), a concept K (for *Keep*) is obtained, representing a contracted version of the original request, such that $K \sqcap S$ is satisfiable w.r.t. $\mathcal{T}$. The solution G to Contraction represents "why" $D \sqcap S$ produces a clash.

Concept Abduction: when R and S are compatible but S does not imply R, Abduction determines what should be hypothesized in S in order to completely satisfy R. The solution H (for *Hypothesis*) to Abduction represents "why" the subsumption relation does not hold. H can be interpreted as *what is requested in R and not specified in S*.

Table 2: $\mathcal{ALN}$ constructors

Name	DL syntax	Manchester syntax
Top	$\top$	`owl:Thing`
Bottom	$\bot$	`owl:Nothing`
Concept	C	*C*
Role	R	*R*
Conjunction	$C \sqcap D$	*C* **and** *D*
Atomic negation	$\neg A$	**not** *A*
Unqualified existential restriction	$\exists R$	R **some** owl:Thing
Universal restriction	$\forall R.C$	R **only** C
Unqualified number restrictions	$\geq n\ R$ $\leq n\ R$	*R* **min** n *R* **max** n
Definition axiom	$A \equiv C$	**Class:** *A* **EquivalentTo:** *C*
Inclusion axiom	$A \sqsubseteq C$	**Class:** *A* **SubClassOf:** *C*

If R and S are incompatible, one can use Contraction to extract the compatible part K and then Abduction to obtain the required H_K for reaching a full match. Furthermore, *penalty functions* can be computed based on the structure and number of concepts in G and H_K [20], defining a well-founded *semantic distance* measure, which can be used to rank a set of resources by relevance (*i.e.*, semantic affinity) w.r.t. the request.

In SWoT contexts, performance constraints of computing devices are strict and require careful software design choices. Adding new constructors makes DL languages more expressive, but leads to an increase in computational complexity of inference services [2]. Hence a tradeoff is needed. This paper refers specifically to the *Attributive Language with unqualified Number restrictions* ($\mathcal{ALN}$) DL. It provides adequate expressiveness, while granting polynomial complexity to both standard and non-standard inference services. $\mathcal{ALN}$ constructors are summarized in Table 2, along with the corresponding *Manchester syntax* serialization[7] of the Web Ontology Language (OWL 2)[8] core Semantic Web standard.

3 Integrating Semantics into Blockchain Frameworks

The proposed approach defines a semantic resource/service discovery layer (without loss of generality, the word resource will be used from now on) built upon a basic blockchain framework. An IoT blockchain is intended as a SOA enabling fundamental operations as registration, discovery, selection and final execution (payment). Such tasks are implemented as

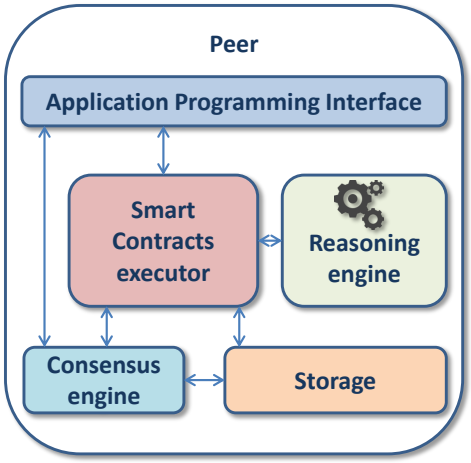

Figure 2: Framework architecture

Smart Contracts (SC), *i.e.*, their accomplishment is distributed and validation is gained by consensus.

As shown in the architectural sketch in Figure 2, the proposed framework is integrated in a standard blockchain system, retaining backward compatibility with existing Application Programming Interfaces (APIs) offered to application developers. Main features are summarized in what follows.

- Agents registered in the blockchain are identified by their *public keys* and associated with *accounts*. Each agent can perform a semantic-based resource discovery in order to take ownership and transfer assets between accounts.
- Each *asset* can be semantically annotated w.r.t. a domain ontology. Agents can register one or more assets by executing a sequence of transactions.
- *Smart contracts* can be semantically enhanced by exploiting the non-standard inference services described in Section 2.2. Particularly, each

[7] OWL 2 Web Ontology Language Manchester Syntax (Second Edition), W3C Working Group Note 11 December 2012, https://www.w3.org/TR/owl2-manchester-syntax/

[8] OWL 2 Web Ontology Language Document Overview (Second Edition), W3C Recommendation 11 December 2012, http://www.w3.org/TR/owl2-overview/

peer locally integrates an embedded matchmaking and reasoning engine, allowing semantic resource discovery [20].

- The semantic-based transactions are tracked and validated transparently by the *consensus engine*.
- Semantic transactions are managed in a *Merkle tree*, an efficient and secure *storage* for detecting erroneous or malicious changes on a transaction block.

4 KNOWLEDGE-BASED BLOCKCHAIN: REACHING SCALABILITY FOR IOT INFRASTRUCTURES

The distinguishing feature of the proposal w.r.t. classical blockchain architectures is the integration of semantic matchmaking fostering the resource discovery. It allows to compare a request with multiple resource descriptions by taking into account semantics of their annotations referred to a shared ontology. As a result, a score is returned which measures the semantic distance between a DL concept expression annotating the request itself and annotations of every available chain resource. This logic-based metric induces a well-founded relevance ranking of resources w.r.t. the request. The inference services also provide a formal *explanation* of discovery outcomes, reinforcing user trust in the discovery process. SOA primitives and corresponding SCs are reported as in what follows.

A. Resource registration. Several resource domains coexist and are tracked in the same blockchain. Every domain is associated to a different ontology, which provides the reference conceptual vocabulary to annotate resources. Every ontology is uniquely identified by a Uniform Resource Identifier (URI), as per OWL specifications. Each node can own a number of resource instances, characterized by the following attributes:

- a URI identifying the resource unambiguously – and possibly representing the resource fruition endpoint, although this is not required;
- a semantic annotation in OWL language describing the resource;
- the URI of the reference ontology;
- a resource price: the proposal is fully compatible with any type of currency unit, pecuniary or otherwise (*e.g.*, in many IoT applications energy or time may be useful currency choices).

In order to make a resource available for discovery and usage, the owner node registers it as an asset on the blockchain-based stream storage, via standard means. For greater efficiency, only the resource URI is stored on the blockchain in the registration transaction.

B. Resource discovery. The proposal adopts a *gossip-based* (a.k.a. *epidemic*) approach [10] to disseminate discovery requests and aggregate results. This grants protocol simplicity and consequently low computational overhead, which is a primary requirement for system scalability. The overall sequence of interaction steps is depicted in Figure 3:

1. The requester randomly selects n nodes and sends a multicast request with the `discover` smart contract. Parameters of the SC are:
   - 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; nodes receiving the request will not process resources annotated w.r.t. other ontologies in the semantic matchmaking;
   - semantic annotation of the request in OWL language, specifying desired resource features and constraints;
   - maximum price p_{max} the requester is willing to pay; resources with a price 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;
   - maximum number of results r_{max} to be returned.
2. Nodes receiving the original request perform two operations in parallel:
   - execute semantic matchmaking of their own resources with the request: for this purpose, resource providers are assumed to be equipped with an on-board lightweight matchmaking engine, implementing the non-standard inference services in Section 2.2. A list of at most r_{max} results is returned, ranked by relevance: each outcome R_i is characterized by: (i) resource owner's public key; (ii) resource URI u_i; (iii) semantic relevance score $s_i \geq s_{min}$; (iv) cost $p_i \leq p_{max}$, in platform currency;
   - select other n nodes randomly and forward the request.

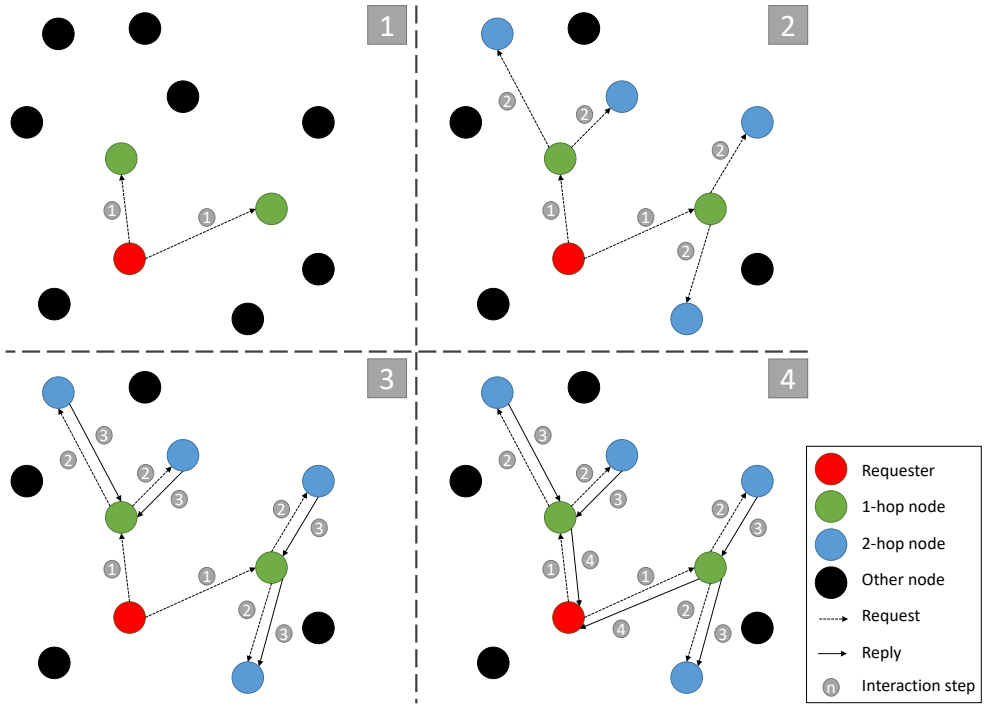

Figure 3: Discovery protocol interaction steps (for $n = 2, m = 2$)

3. Nodes receiving the forwarded requests behave in the same way. When a search depth threshold m is reached (in hops from the original requester), nodes do not forward the request any further and just perform matchmaking locally.

4. Each queried node returns results to the sender, which propagates them back to the original requester following the same route of the requests. In this way, each request will reach $\sum_{i=1}^{m} n^i$ random nodes, with n and m tunable parameters: in the current implementation they have been chosen *a priori* at global level, but –with the proposed infrastructure already in place– it is trivial to implement them as variable on a node-by-node basis and/or dynamically adaptable in order to maximize application-specific performance goals.

C. Explanation. The invocation of the `explain` SC is optional in a typical resource discovery workflow. It is used when a requester needs a justification of the matchmaking outcome for a specific resource among received results. This can be useful, *e.g.*, to trigger a request refinement process [18]. Parameters of the SC are (i) the semantic annotation of the request and (ii) the URI of the discovered resource. The recipient replies with matchmaking outcome explanation, structured as: (i) semantic affinity score $0 \leq s_i \leq 1$; concept expressions of G and K from Concept Contraction and of H from Concept Abduction. If computational capabilities allow it, the resource owner can host a local cache to store temporarily the matchmaking outcomes from executed `discover` calls, in order to avoid computing them again –even though for just one resource– in case of explanation request.

D. Resource selection. After receiving all results – or just a subset, if the response delay of some nodes is greater than a fixed *timeout*– the requester selects the best resource(s) with the `select` smart contract, sending a unicast message to the resource owner with the resource URI and contextually a currency payment. The recipient answers with a properly usable resource representation, which depends on the actual kind of resource and meaning of the URI, *e.g.*, an interface endpoint to access a networked device or a further SC to be invoked. The proposal does not constrain resource fruition in any way, leaving application-specific details to the semantic annotation of the resources themselves.

Resource discovery, explanation and selection transactions are recorded on the blockchain for robustness, traceability and accountability purposes.

5 Case Study: Variable Scale IoT for Factory Automation

Industry 4.0 is a key emerging sector of the IoT, which can greatly benefit from blockchain technologies [6]. Notably, in the pharmaceutical industry blockchain offers not only a decentralized collaboration infrastructure, but also a ledger for manufacturing process traceability and prevention of product counterfeiting [15]. A case study in distributed and dynamic scheduling of production tasks in a pharmaceutical factory is presented to better clarify capabilities of the proposed framework, also highlighting its benefits.

Let us consider a basic example where every production task is accomplished in a fully autonomous environment by agents –representing production lines– driven by semantic-based reasoning. All agents are modeled as blockchain nodes and are able to share assets. In the proposed approach, every asset is a specific service with a semantic annotation, accomplishing a production step in a certain way and with given preconditions and effects. Assets are also associated to a pecuniary price; in the following examples, prices are normalized w.r.t. production quantity. Similar services can be exposed by different providers for different prices, creating a suitable SOA-based marketplace. A provider node can offer its services if it is not already busy in another production run.

When a new order is submitted, distributed scheduling occurs, aimed at covering the whole production process through semantic-based discovery and composition of sequences of elementary tasks. Each task is modeled with two semantic annotations, for *preconditions* and *effects* respectively. Every single discovery step retrieves a logic-based ranking of available service instances for a required production task. Additionally, the most suitable service can be selected in a fully autonomous way, according to the current availability of providers and the price. Thus, the overall scheduling process is performed in a top-down way: firstly, discovery finds the service(s) with the final requested output as effects; in turn, nodes providing those services will discover other providers to satisfy their required preconditions, until services are found which have no uncovered preconditions. This allows step-wise distributed discovery of a service chain to be executed (in reverse order) to get the requested product, all without human intervention.

The initial input consists of the semantic annotation of production request and a total currency budget. The given preconditions which must be satisfied by the production task are compared with the effects of available services, exploiting the non-standard inferences described in Section 2.2 and computing a semantic relevance score. The preconditions of the best service become the new input for the next discovery step. This process is carried out iteratively until it reaches the first step of production. For the sake of simplicity, in the subsequent example the availability of raw materials is assumed as an always met precondition. At each step, the total budget is reduced by the price of the selected service. If during the process the budget becomes negative, backtracking is performed by selecting a less expensive service, even though with lower semantic affinity. When the overall discovery ends, the dynamic scheduling of the production process is accomplished.

Table 3: Results of semantic matchmaking w.r.t. initial request R

Service	Annotation	Semantic relevance
ASA Regimer	E_{13}	1.0
ASA Regular	E_{12}	0.96
ASA Extra	E_{11}	0.95

The following example focuses on the specific production of medications based on acetylsalicylic acid (ASA, a.k.a. aspirin). Different ASA concentrations correspond to different therapeutic uses, ranging from low-dose maintenance therapy to prevent recurring heart attacks or stroke, to regular or extra dose for normal or acute pain relief, respectively.

A private hospital requires the supply of a batch of ASA-based medications, to be used for therapeutic purposes in elderly heart patients for secondary prevention after an infarction event. The desired concentration for such therapy is between 75 and 100 mg. The maximum available monetary budget is €100 per production batch. Using concepts defined in the domain ontology –a relevant excerpt of it is shown in Figure 4– the request R is formalized as reported in Figure 5 in OWL 2 Manchester Syntax. It is the input of the first semantic-based discovery step. Figure 5 also presents available services and their prices in the form $S_{ji} \equiv \langle E_{ji}, P_{ji}, p_{ji} \rangle$, where j denotes the production step and i the specific service instance. Effects E_{ji} and preconditions P_{ji} are described w.r.t. the same domain ontology.

Let us suppose a semantic affinity threshold s_{min} of 0.9: only services with at least this score will be considered. The first semantic matchmaking returns the ranked list of services in affinity order reported in Table 3. Overall match score s ranges from 0 to 1. It is computed by means of the formula:

$$f(x) = \begin{cases} 1-p, & 1-p > s_{min} \\ 0, & otherwise \end{cases} \quad (1)$$

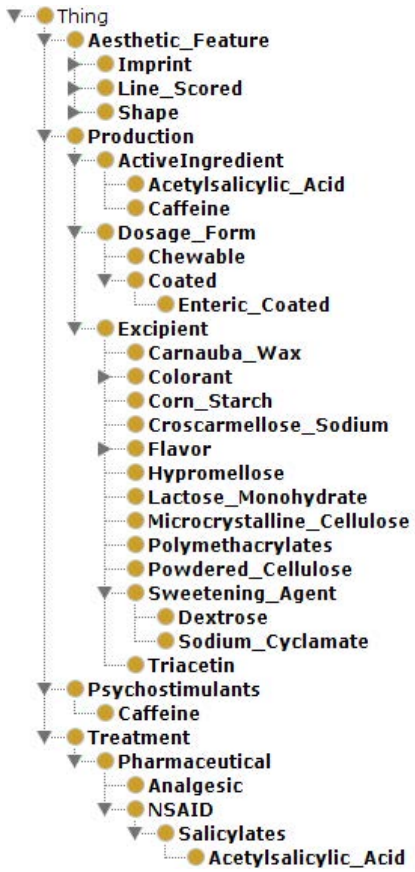

Figure 4: Excerpt of the ontology engineered for the case study

with the semantic penalty function p computed as:

$$p = \frac{w \cdot penalty_c + (1 - w) \cdot penalty_a}{penalty_{max}} \qquad (2)$$

Where $penalty_c$ is the penalty calculated by Concept Contraction between the request R and each annotation of service effects E_i, while $penalty_a$ is the penalty value of the Concept Abduction procedure between the consistent part K of the request R and E_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 $\top$ concept, and depends only on axioms in the reference domain ontology [18]). The parameter w ranges from 0 to 1: it determines the relative weight of explicitly conflicting elements in E_i w.r.t. R. The chosen value in the following example is 0.6, to penalize services having conflicting features with requests.

The first semantic matchmaking step aims to select the medicine, which is distinguished essentially by the active ingredient concentration. The hospital node on the blockchain invokes the `discover`

Request: $R \equiv$ (hasActiveIngredient **only** Acetylsalicylic_Acid) **and** (hasASAConcentration **min** 75) **and** (hasASAConcentration **max** 100)

S_{11}: *Acetylsalicylic_Acid_Extra* $\equiv \langle E_{11}, P_{11}, p_{11} \rangle \equiv \langle \{$(hasActiveIngredient **min** 1) **and** (hasActiveIngredient **only** (Acetylsalicylic_Acid **and** Caffeine) **and** (hasASAConcentration **min** 500) **and** (hasASAConcentration **max** 500)$\}$, $\{$(hasDosageForm **min** 1) **and** (hasDosageForm **only** Coated)$\}$, €20$\rangle$

S_{12}: *Acetylsalicylic_Acid_Regular* $\equiv \langle E_{12}, P_{12}, p_{12} \rangle \equiv \langle \{$(hasActiveIngredient **min** 1) **and** (hasActiveIngredient **only** Acetylsalicylic_Acid) **and** (hasASAConcentration **min** 325) **and** (hasASAConcentration **max** 325)$\}$, $\{$(hasDosageForm **min** 1) **and** (hasDosageForm **only** Chewable)$\}$, €25$\rangle$

S_{13}: *Acetylsalicylic_Acid_Regimer* $\equiv \langle E_{13}, P_{13}, p_{13} \rangle \equiv \langle \{$(hasActiveIngredient **min** 1) **and** (hasActiveIngredient **only** Acetylsalicylic_Acid) **and** (hasASAConcentration **min** 81) **and** (hasASAConcentration **max** 81)$\}$, $\{$(hasDosageForm **min** 1) **and** (hasDosageForm **only** Enteric_Coated)$\}$, €30$\rangle$

Figure 5: Initial request and services in the first matchmaking step

Table 4: Results of semantic matchmaking w.r.t. request P_{13}

Service	Annotation	Semantic relevance
Enteric Coated Caplet	E_{21}	1.0
Coated Tablet	E_{23}	0.93
Coated Caplet	E_{22}	0.93
Chewable Tablet	E_{24}	0.86

smart contracts on suppliers' nodes providing finished products. The product best matching the request is *Acetylsalicylic_Acid_Regimer* and the budget is updated to €70.

When manufacturing medicines for daily use in maintenance therapy, it is good practice to adopt a capsule dosage form with enteric coating to prevent gastric irritation. In order to discover the most suitable provider for medicine dosage form, a new semantic request is submitted to available nodes. The request annotation corresponds to the preconditions of the previously selected service, as depicted in Figure 6. Results are shown in Table 4. The most suitable dosage form is *Enteric_Coated_Caplet*. On the other hand, *Chewable_Tablet* will never be considered because its score is less than the affinity threshold. The price of service S_{21} shall be deducted from budget, leaving €35.

In the selection of excipients guaranteeing the properties associated with the dosage form, the medicine release profile must be considered. Release must occur in the intestinal tract rather than in the stomach. By joining

Request: P_{13}

S_{21}: $Enteric\_Coated\_Caplet \equiv \langle E_{21}, P_{21}, p_{21} \rangle$ $\equiv$ ⟨{(hasDosageForm **min** 1) **and** (hasDosageForm **only** Enteric_Coated) **and** (hasImprint **min** 1) **and** (hasImprint **only** One_Side_Active_Ingredient_Dosage) **and** (hasShape **min** 1) **and** (hasShape **only** Round)}, {(hasExcipient **min**1) **and** (hasExcipient **only** (Hypromellose **and** Polymethacrylates))}, €30⟩

S_{22}: $Coated\_Caplet \equiv \langle E_{22}, P_{22}, p_{22} \rangle \equiv$ ⟨{(hasDosageForm **min** 1) **and** (hasShape **min** 1) **and** (hasDosageForm **only** Coated) **and** (hasImprint **min** 1) **and** (hasImprint **only** One_Side_Logo) **and** (hasLineScored **min** 1) **and** (hasLineScored **only** One_Side_Line_Scored) **and**(hasShape **only** Caplet)}, {(hasExcipient **min** 1) **and** (hasExcipient **only** Hypromellose)}, €25⟩

S_{23}: $Coated\_Tablet \equiv \langle E_{23}, P_{23}, p_{23} \rangle \equiv$ ⟨{(hasDosageForm **min** 1) **and** (hasDosageForm **only** Coated) **and** (hasImprint **min** 1) **and** (hasImprint **only** Both_Side_Logo) **and** (hasShape **min** 1) **and** (hasShape **only** Round)}, {(hasExcipient **min** 1) **and** (hasExcipient **only** Hypromellose)}, €25⟩

S_{24}: $Chewable\_Tablet \equiv \langle E_{24}, P_{24}, p_{24} \rangle \equiv$ ⟨{(hasDosageForm **some**) **and** (hasDosageForm **only** Chewable) **and** (hasImprint **some**) **and** (hasImprint **only** One_Side_Logo) **and** (hasShape **some**) **and** (hasShape **only** Round)}, {(hasExcipient **some**) **and** (hasExcipient **only** (Flavor **and** Sweetening_Agent)}, €30⟩

Figure 6: Request and services in the second matchmaking step

Request: P_{21}

S_{31}: $Orange\_Chewable\_Tablet\_Excipients \equiv \langle E_{31}, P_{31}, p_{31} \rangle \equiv$ ⟨{(hasExcipient **min** 1) **and**(hasExcipient **only** (Corn_Starch **and** Dextrose **and** FDC_yellow_6_aluminum_lake **and** Orange_Flavor **and** Sodium_Cyclamate)}, {owl:Thing}, €25⟩

S_{32}: $Cherry\_Chewable\_Tablet\_Excipients \equiv \langle E_{32}, P_{32}, p_{32} \rangle \equiv$ ⟨{(hasExcipient **min** 1) **and** (hasExcipient **only** (Cherry_Flavor **and** Corn_Starch **and** Dextrose **and** FDC_red_3_aluminum_lake **and** Sodium_Cyclamate)}, {owl:Thing}, €30⟩

S_{33}: $Delayed\_Release\_Tablet\_Excipients \equiv \langle E_{33}, P_{33}, p_{33} \rangle \equiv$ ⟨{(hasExcipient **min** 1) **and** (hasExcipient **only**(Carnauba_Wax **and** Corn_Starch **and** Croscarmellose_Sodium **and** Hypromellose **and** Lactose_Monohydrate **and** Microcrystalline_Cellulose **and** Polymethacrylates)}, {owl:Thing}, €35⟩

S_{34}: $Coated\_Tablet\_Excipients \equiv \langle E_{34}, P_{34}, p_{34} \rangle \equiv$ ⟨{(hasExcipient **min** 1) **and** (hasExcipient **only** (Corn_Starch **and** Hypromellose **and** Powdered_Cellulose **and** Triacetin)},{owl:Thing}, €25⟩

Figure 7: Request and services in the last matchmaking step

Table 5: Results of semantic matchmaking w.r.t. request P_{21}

Service	Annotation	Semantic relevance
Delayed Release Tablet Excipients	E_{33}	1.0
Coated Tablet Excipients	E_{34}	0.93
Orange Chewable Tablet Excipients	E_{31}	0.87
Cherry Chewable Tablet Excipients	E_{32}	0.83

the coating agent (e.g., *hypromellose) with specific film-forming agents* (e.g., *polymethacrylates), it is possible to release in specific intestinal sites.* Figure 7 shows annotation P_{21}, which has become the new request for the next `discover` SC calls to excipient suppliers, and available semantic service descriptions.

As summarized in Table 5, the best semantic affinity score is given by $Delayed\_Release\_Tablet\_Excipients$ effects. $Coated\_Tablet\_Excipients$ missed the requested polymethacrylates excipient, while chewable tablets missed also hypromellose. The final budget is €0, so allocated currency is enough to cover the overall production process. Production can now start in a completely automatic way, by invoking `select` SCs in the reverse order of the semantic-based composition process.

Notice that semantic annotations in the above examples have been simplified for the sake of understandability, as evident by the fact that relevance scores are generally high. In real blockchain-based industrial IoT scenarios, more complex requests and service descriptions are expected, and semantic-based discovery will be even more important to rank resources based on their specific features.

6 Experiments

This section collects and presents early experimental results devoted to assess effectiveness and scalability of the proposed approach. Implementation and performance evaluation were carried out exploiting the *Iroha* framework of Hyperledger. Iroha is implemented in C++ and provides the following components:

- *Iroha core* includes the distributed ledger infrastructure, the consensus algorithm, the peer-to-peer communication and the smart contract environment. Particularly, the Sumeragi BFT consensus algorithm is used, inspired by B-Chain [5].

- *Native iOS/Android libraries* provide functions for interacting with the blockchain (*e.g.*, digitally signing transactions).

The developed prototype provided the following enhancements w.r.t. the basic Iroha environment:

- The server API was extended with support for semantic matchmaking.
- The SCs described in Section 4 were implemented in Java. The *Mini-ME* reasoning and matchmaking engine [20] was integrated to execute inference tasks.
- *zlib*[9] compression library was exploited to cope with the well-known verbosity of ontology languages such as OWL.

Unfortunately, comparative evaluations could not be carried out, as no other blockchain-based resource discovery approaches have been found in literature or in the market. Furthermore, so far semantic-based Smart Contract approaches exist as theoretical proposals, as discussed in Section 7, but they have not been integrated yet in blockchain implementations.

Materials and methods. In order to obtain a quantitative performance analysis of the proposed framework, various parameters were measured and evaluated. Small, medium and large scale scenarios were considered, respectively with 10, 50 and 150 nodes. In each scenario nodes were split in two sets: *producers*, *i.e.*, providers of annotated resources, registered in the blockchain; *consumers*, *i.e.*, resource requesters which execute smart contracts to perform semantic-based discovery, as described in Section 4.

The following parameters were set: (i) experiment duration was 300 s; (ii) the consumer/producer ratio was 0.1; (iii) each producer registered 20 randomly-generated annotations; (iv) each consumer sent a new randomly-generated request every 10 s; (v) each request could be forwarded to a subset of four nodes; (vi) a request was aborted if no match was found after the second hop; (vii) the minimum threshold of semantic affinity was 0.9.

Each scenario was executed several times by varying the following parameters: (i) the discovery timeout was set to 2, 6 and 10 s; (ii) the explanation SC, described in Section 4 was either enabled or disabled. Experiments were performed on a workstation with Intel Xeon E5-2643 CPU at 3.30 GHz, 48 GB of RAM and Ubuntu 16.04 (64bit) operating system.

The Docker platform was used to deploy the testbed, by performing the following steps:

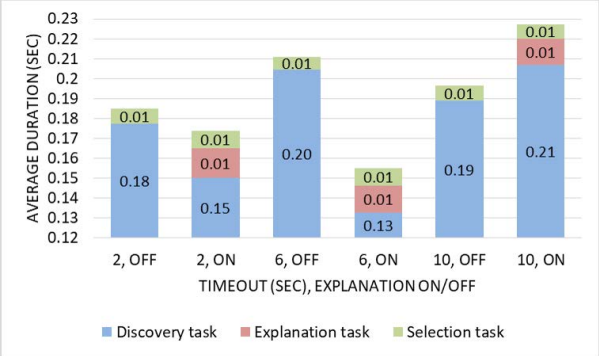

Figure 8: Processing time for tasks on 10 nodes

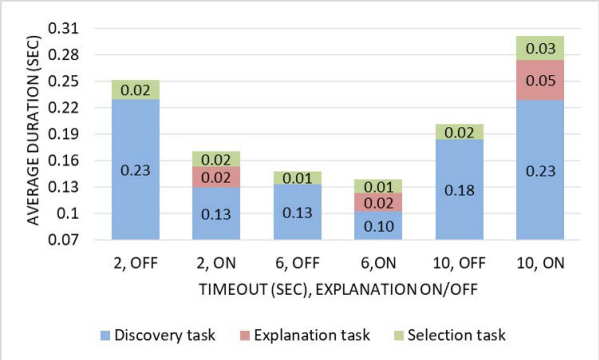

Figure 9: Processing time for tasks on 50 nodes

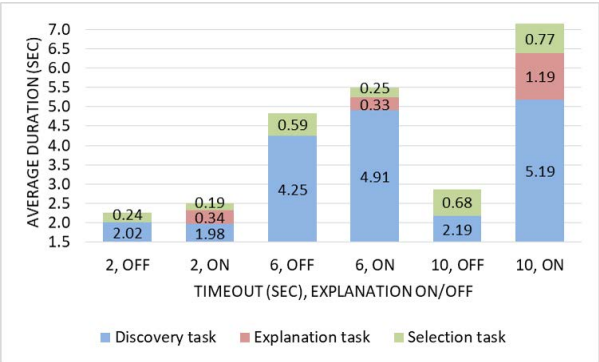

Figure 10: Processing time for tasks on 150 nodes

- the prototype was compiled as a Docker image, used to create all the scenarios;
- each node was executed as a container instance of the compiled image;
- a Docker network was created to allow communications among the nodes;
- the Docker API SDK[10] was used to manage the experiments execution.

[9] http://zlib.net/

[10] https://github.com/spotify/docker-client

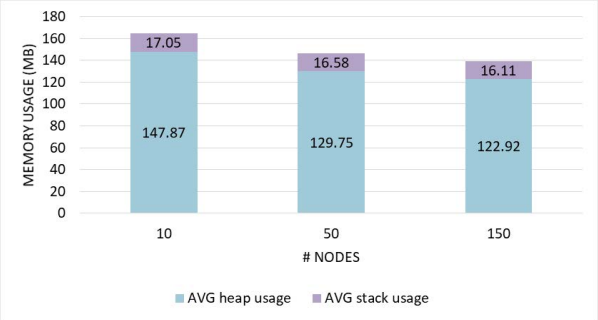

Figure 11: JVM average memory usage per node

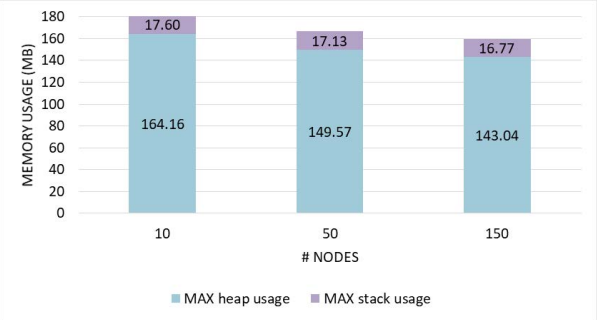

Figure 12: JVM maximum memory usage per node

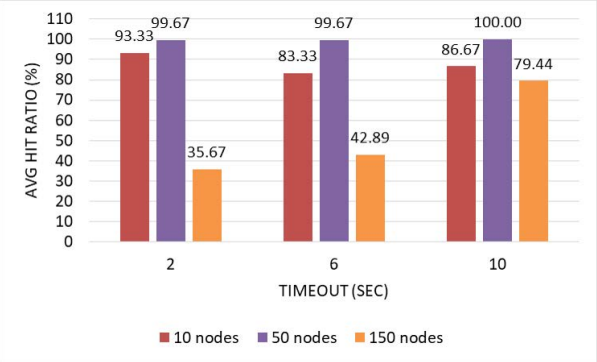

Figure 13: Average hit ratio depending to timeout

The following statistics were collected: (i) average processing time for computing tasks on a given request; (ii) average turnaround time for accomplishing requests; (iii) average Java Virtual Machine (JVM) memory usage per node; (iv) average hit ratio per node. Hit ratio is defined here as the percentage of requests which retrieve at least one resource satisfying both price and semantic relevance constraints within the given timeout. Experimental results are reported hereafter.

Time. There were no substantial differences between average turnaround time for requests in experiments involving 10 and 50 nodes, as Figure 8 and Figure 9 show. Absolute time values can be deemed as very low with the small and medium scenarios, while with 150 nodes turnaround time reached the timeouts at 2 and 6 seconds, as depicted in Figure 10. Processing times tend to increase at higher scales due to the needed consensus about computation results among a larger number of nodes. Furthermore, in all the experiments the time of discovery process dominates explanation and selection. This was expected, as semantic matchmaking is the most computationally intensive task, even if performed with an optimized reasoning engine [20].

Memory. RAM consumption per node is mainly related to Java Virtual Machine requirements for the *Iroha* framework. No RAM constraints were imposed in the configuration of the JVM and Docker. The main reason for that was to avoid affecting other performance metrics. As reported in Figure 11 and Figure 12, the average and maximum memory usage per node decreased slightly for larger scenarios. This can be explained with memory management policies of the testbed host, which must divide RAM among an increasing number of nodes.

Hit ratio. Figure 13 shows average hit ratio is closely related to the number of nodes. The best results were obtained in the 50 nodes scenario. The 10 nodes scenario had a lower average hit ratio, likely due to fewer semantic resources available on the whole network, making it more difficult to retrieve a resource satisfying both price and semantic affinity constraints. Finally, in the 150 nodes scenario the average hit ratio was clearly lower. In part, this was due to Docker resource contention issues affecting the CPU, file system and network when the number of containers on the same host grew (as Iroha nodes are rather resource-consuming processes). This compounded overhead with the inherent complexity of consensus algorithms, leading to the turnaround time issues already discussed. The effect was an increased probability of timeout expiration and consequent resource miss. In Figure 13, indeed, it can be noticed that with 150 nodes the larger the timeout the higher the hit ratio.

Globally, experimental outcomes show the approach is effective and sustainable for small-to-medium permissioned blockchains. Higher scalability can be achieved by properly setting the discovery protocol parameters concerning breadth and width of request propagation as well as response timeout, based on the expected number of participating nodes.

By comparing the relative duration of resource discovery and selection sub-tasks, it can be inferred that the semantic-based proposed enhancement causes an increase of about an order of magnitude. This may be considered high, but the case study in Section 5 demonstrates how many advanced

blockchain applications and scenarios in the IoT will be impossible without the intelligence and flexibility of service/resource discovery provided by a sophisticated approach.

Larger-scale scenarios could not be set up on the reference testbed due to the above issues with Docker. A new testbed based on a computer cluster will be set up to re-evaluate semantic-enhanced blockchain performance in the same scenarios –possibly removing the single-host bias– as well as to run larger networks.

7 Related Work

Centralized information, service and device management models are clearly not scalable enough for the ever-growing IoT. In addition to cost and performance issues, they pose security and trust problems. In [17] the transparent trustless peer-to-peer models enabled by blockchain technologies are proposed as a viable approach to sustain the current and future expansion of reliable IoT networks and applications. Emerging distributed file systems, billing services and other blockchain-based tools can be leveraged as an application-agnostic machine-to-machine middleware layer for running IoT resource/service marketplaces with minimal or no human intervention [3].

Industrial research is experimenting with many possible scenarios within Industry 4.0, deemed as a prominent IoT-based use case for blockchain [6]. Asset tracking and supply chain are among the most popular applications, due to the widely recognized benefits of trustless DAO collaboration [13, 14, 15] and the easy fit of blockchain solutions in existing industry standards for information-sharing distributed infrastructures, most notably the Electronic Product Code Information Services (EPCIS) [11]. The simplest approaches rely on transactional ledgers for asset transfer, which grant high troughput with low costs [3]. Blockchain networks based on SCs enable more flexible systems, allowing any application logic to be implemented and embedded in the blockchain [3], and also supporting discoverable, composable and verifiable multi-step business processes in multi-party service-oriented architectures (SOA) [16]. The proposed approach clearly falls in this category.

Unfortunately, the benefits of smart contracts come at a not-negligible cost in terms of concurrent execution of transactions and, consequently, system throughput [3]. This occurs because in the general case, before executing a smart contract, a node cannot know what computing resources it will need –even possibly including other smart contracts– and what its effects will be on the system state. That makes it impossible to run all transactions in a block in parallel. In Bitcoin-style asset transfer blockchains, on the contrary, transaction have a fixed inherent semantics: conditions for transaction dependencies and ordering are simpler and known in advance, leading to the possibility to execute (usually most of) the transactions in a block concurrently and thus achieve higher throughput and scalability. This is a significant barrier to advanced blockchain applications in the IoT, which require freely specifiable business logic and low-latency high-throughput transactions at the same time.

Research on blockchain scalability is very active, mainly by optimizing performance of consensus protocols [22] and by introducting parallelism in a blockchain through *sidechains* and/or *sharding* [4]. Basically, the use of sidechains will transform the chain structure in a direct acyclic graph. On the other hand, sharding 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. Research results, however, are not mature enough [23] for building efficient, robust, large-scale IoT-oriented blockchains. In this respect, the approach proposed here aimed to mitigate scalability issues by providing a semantic-enabled SOA above contract-based blockchains: while enabling general-purpose service/resource discovery and retrieval, this layer has a finite set of smart contracts as primitives, which do not have recursive calls.

Many opportunities exist for exploiting logic-based technologies in blockchains. In [6] a prototypical ontology was proposed to annotate transactions with Linked Data [7] in order to make contents of the blockchain easier to explore for humans through semantic-enabled user agents. The ontology-based smart contract design of a proof-of-concept blockchain system in [12] enabled traceability in supply chains. Besides acting as design guidelines, logical languages can be used to specify formally and execute smart contracts. Several logical frameworks have been applied to the problem of SC specification and execution. The work in [9] used *defeasible reasoning*, a well-known approach to formalize legal regulations and contracts. On the other hand, [8] endorsed the usage of *Linear Temporal Logic* (LTL), which is implemented in a large number of model checking systems. This allows formal verification that the behavior of a SC satisfies specific conditions. The need for formal verification of SCs in business-oriented blockchains was also highlighted in [16].

In the present work DLs are used to enable discovery and composition of services/resources on a blockchain. This is a relevant issue in blockchain-based marketplaces, and particularly in upcoming IoT applications [8]. To the best of our knowledge no

other automated semantic-based discovery approach for blockchain systems exists; we claim it has the potential of a clear improvement in flexibility and quality of discovery w.r.t. existing discovery approaches mutuated from the Domain Name System (DNS) [3].

8 Conclusion and Future Work

Given typical scalability issues in large and very large IoT infrastructures, the paper proposed a framework redesigning resource discovery thanks to the basic blockchain infrastructure. Registration, discovery, selection and finalizing operations have been revisited as smart contracts in order to comply with an opportunistic and distributed execution leveraging validation by consensus. An important feature of the proposal resides on the *logic-based explanation* of discovery outcomes, obtained through non-standard inference for matchmaking among request and resources. Early experiments on the *Hyperledger Iroha* framework have been carried out and have been presented to assess feasibility of the approach.

Future work will be essentially directed to migrate the testbed toward the *Docker Swarm* scheduling tool in cluster computing environments, in order to increase the simulation scale of several order of magnitude nodes.

Acknowledgments

The authors acknowledge partial support of Apulia region project PERSON (PERvasive game for perSOnalized treatment of cognitive and functional deficits associated with chronic and Neurodegenerative diseases).

References

[1] F. Baader, D. Calvanese, D. Mc Guinness, D. Nardi, and P. Patel-Schneider, *The Description Logic Handbook*. Cambridge University Press, 2002.

[2] R. Brachman and H. Levesque, "The Tractability of Subsumption in Frame-based Description Languages," in *4th National Conference on Artificial Intelligence (AAAI-84)*, 1984, pp. 34–37.

[3] K. Christidis and M. Devetsikiotis, "Blockchains and Smart Contracts for the Internet of Things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[4] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. G. Sirer *et al.*, "On scaling Decentralized Blockchains," in *International Conference on Financial Cryptography and Data Security*, 2016, pp. 106–125.

[5] S. Duan, H. Meling, S. Peisert, and H. Zhang, "BChain: Byzantine Replication with High Throughput and Embedded Reconfiguration," in *Principles of Distributed Systems - 18th International Conference, OPODIS 2014*, December 2014, pp. 91–106.

[6] M. English, S. Auer, and J. Domingue, "Block Chain Technologies & The Semantic Web: A Framework for Symbiotic Development," in *Computer Science Conference for University of Bonn Students*, J. Lehmann, H. Thakkar, L. Halilaj, and R. Asmat, Eds., 2016, pp. 47–61.

[7] T. Heath and C. Bizer, *Linked data: Evolving the Web into a Global Data Space*, ser. Synthesis lectures on the semantic web: theory and technology. Morgan & Claypool Publishers, 2011.

[8] R. Hull, V. S. Batra, Y.-M. Chen, A. Deutsch, F. F. T. Heath III, and V. Vianu, "Towards a Shared Ledger Business Collaboration Language based on Data-Aware Processes," in *International Conference on Service-Oriented Computing*, 2016, pp. 18–36.

[9] F. Idelberger, G. Governatori, R. Riveret, and G. Sartor, "Evaluation of Logic-Based Smart Contracts for Blockchain Systems," in *International Symposium on Rules and Rule Markup Languages for the Semantic Web*, 2016, pp. 167–183.

[10] M. Jelasity, A. Montresor, and O. Babaoglu, "Gossip-Based Aggregation in Large Dynamic Networks," *ACM Transactions on Computer Systems (TOCS)*, vol. 23, no. 3, pp. 219–252, 2005.

[11] A. Kennedy, M. Southall, G. Morgan, K. Traub *et al.*, "EPC Information Services (EPCIS) Specification," GS1, Tech. Rep., May 2014, available: http://www.gs1.org/epcis/epcis/1-1.

[12] H. M. Kim and M. Laskowski, "Towards an Ontology-Driven Blockchain Design for Supply Chain Provenance," 2016, https://ssrn.com/abstract=2828369.

[13] K. Korpela, J. Hallikas, and T. Dahlberg, "Digital Supply Chain Transformation toward Blockchain Integration," in *Proceedings of the 50th Hawaii International Conference on System Sciences*, 2017, pp. 4182–4191.

[14] J. Mattila, T. Seppälä, and J. Holmström, "Product-centric Information Management: A Case Study of a Shared Platform with Blockchain Technology," in *Berkeley Roundtable on the International Economy*, 2016.

[15] M. Mettler, "Blockchain Technology in Healthcare: The Revolution Starts Here," in *IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom)*, 2016, pp. 1–3.

[16] A. Norta, "Creation of Smart-Contracting Collaborations for Decentralized Autonomous Organizations," in *International Conference on Business Informatics Research*. Springer, 2015, pp. 3–17.

[17] V. Pureswaran and P. Brody, "Device Democracy: Saving the Future of the Internet of Things," IBM Institute for Business Value, Tech. Rep., September 2014, available: http://www-935.ibm.com/services/us/gbs/thoughtleadership/internetofthings.

[18] M. Ruta, E. Di Sciascio, and F. Scioscia, "Concept Abduction and Contraction in Semantic-based P2P Environments," *Web Intelligence and Agent Systems*, vol. 9, no. 3, pp. 179–207, 2011.

[19] F. Scioscia and M. Ruta, "Building a Semantic Web of Things: Issues and Perspectives in Information Compression," in *Semantic Web Information Management (SWIM'09). In Proceedings of the 3rd IEEE International Conference on Semantic Computing (ICSC 2009)*, 2009, pp. 589–594.

[20] F. Scioscia, M. Ruta, G. Loseto, F. Gramegna, S. Ieva, A. Pinto, and E. Di Sciascio, "A Mobile Matchmaker for the Ubiquitous Semantic Web," *International Journal on Semantic Web and Information Systems (IJSWIS)*, vol. 10, no. 4, pp. 77–100, 2014.

[21] N. Szabo, "Formalizing and Securing Relationships on Public Networks," *First Monday*, vol. 2, no. 9, 1997.

[22] M. Vukolić, "The quest for Scalable Blockchain Fabric: Proof-of-Work vs. BFT Replication," in *International Workshop on Open Problems in Network Security*. Springer, 2015, pp. 112–125.

[23] H. Wang, K. Chen, and D. Xu, "A Maturity Model for Blockchain Adoption," *Financial Innovation*, vol. 2, no. 12, 2016.

AUTHOR BIOGRAPHIES

Michele Ruta received the master's degree in Electronics Engineering from the Polytechnic University of Bari in 2002 and the Ph.D. in Computer Science in 2007. He is currently associate professor. His research interests include pervasive computing and ubiquitous web, mobile e-commerce applications, knowledge representation systems and applications for wireless ad-hoc contexts. On these topics, he has co-authored more than 100 papers in international journals, edited books and conferences. His papers at the ICEC-2007 and SEMAPRO-2010 conferences received the best paper award. He is involved in various research projects, he is Program Committee member of several international conferences and workshops and editorial board member of journals in areas related to his research interests.

Floriano Scioscia received the master's degree in Information Technology Engineering with honours from Polytechnic University of Bari in 2006, and the Ph.D. in Information Engineering in 2010 from the same institution. He is currently a post-doc research fellow at the Information Systems Laboratory (SisInfLab) in the same university. His research interests include knowledge representation systems and applications for pervasive computing and the Internet of Things. He co-authored about 70 papers in international journals, edited books and conferences, and received the best paper award at the ICEC-2007 and SEMAPRO-2010 Conferences.

Saverio Ieva received the master's degree in Information Technology Engineering from Polytechnic University of Bari in 2007. He is currently a research assistant at the Information Systems Laboratory (SisInfLab) in the same university. His research interests include pervasive computing and the Internet of Things, knowledge representation systems and applications for navigation systems. On these topics, he has co-authored papers in international journals and conferences.

Giovanna Capurso received the masters degree in Information Technology Engineering with honours from Polytechnic University of Bari in 2016. She is currently a Ph.D. student in Electrical and Information Engineering in the same institution. Her research interests include knowledge representation systems, reasoning and applications for pervasive computing and ubiquitous smart environments.

Eugenio Di Sciascio received the master's degree with honours from University of Bari, and the Ph.D. from Polytechnic University of Bari. He is currently full professor of Information Systems at the same university, where he is serving as Rector. He leads the research group of the Information Systems Laboratory. Formerly, he has been an assistant professor at University of Lecce and associate professor at Polytechnic University of Bari. His research interests include multimedia information retrieval, knowledge representation and e-commerce. He is involved in several national and European research projects related to his research interests. He co-authored papers that received best paper awards at the ICEC-2004, IEEE CEC-EEE-2006, ICEC-2007, SEMAPRO-2010 and ICWE-2010 conferences.