Hyperledger Fabric Blockchain as a Service for the IoT: Proof of Concept

Chapter · October 2019

DOI: 10.1007/978-3-030-32065-2_12

6 authors, including:

Sasa Pesic
University of Novi Sad
9 PUBLICATIONS **8** CITATIONS
SEE PROFILE

Milos Radovanovic
University of Novi Sad
87 PUBLICATIONS **981** CITATIONS
SEE PROFILE

Mirjana Ivanovic
University of Novi Sad
401 PUBLICATIONS **2,708** CITATIONS
SEE PROFILE

Milenko Tosic
La Citadelle Inzenjering
18 PUBLICATIONS **37** CITATIONS
SEE PROFILE

Hyperledger Fabric Blockchain as a Service for the IoT: Proof of Concept

Saša Pešić[1], Miloš Radovanović[1], Mirjana Ivanović[1], Milenko Tošić[2], Ognjen Iković[2], and Dragan Bošković[2]

[1] University of Novi Sad, Faculty of Sciences, Serbia
{sasa.pesic,radacha,mira}@dmi.uns.ac.rs
[2] VizLore Labs Foundation, Serbia
{milenko.tosic,ognjen.ikovic,dragan.boskovic}@vizlore.com

Abstract. Blockchain as a Service for the Internet of Things is an emerging topic in the blockchain research and industrial community, especially relating to increased system consistency, security, and privacy. Blockchains represent highly distributed and autonomous decision-making systems with distributed data and process management. Internet of Things systems share these characteristics, while also bringing the cyber-physical dimension and machine-to-machine interaction concept to the ecosystem of devices and users. Blockchain infrastructure setup and smart contract prototyping are cumbersome tasks. Onboarding an existing system to a blockchain network takes a significant amount of time and manual effort, and incorporating business logic requires a series of complex steps. For IoT systems, these task needs to be carried out having in mind the typical characteristics of such systems: low hardware, storage, and networking capabilities and high node churn and transaction volume. Moreover, these tasks need to be semi to fully automated in terms of workflows that support easy-to-use integration mechanisms for onboarding of diverse IoT infrastructures and on-demand business logic generation. In this paper, we present a Hyperledger Fabric-based Blockchain as a Service for addressing the identified challenges. Specifically, the framework is tailored to answer to specific requirements of IoT systems through three major services: Hyperledger Fabric Infrastructure Configuration Generator, Hyperledger Fabric Chaincode Builder and Hyperledger Fabric Operator Modules.

Keywords: Blockchain · Internet of Things · Distributed systems

1 Introduction

Driven by artificial intelligence, cognitive computing and new solutions for device-to-device connectivity as well as rising technologies concerning big data and data analytics, the adoption of the Internet of Things (IoT) is accelerating rapidly. IoT applications are distributed by nature. Hence, distributed ledger technology (DLT) such as blockchain has the potential to play a big role in how devices communicate in IoT. Since blockchains are designed as systems that provide the basis for transaction processing or semantically richer smart contracts, they can be leveraged to model IoT processes. This way, blockchains can improve not just data integrity, confidentiality, and compliance in the IoT, but also enhance IoT features and cost-efficiency.

Integrating a full blockchain framework into an existing industrial IoT system is a cumbersome task. It requires setting up an entirely separate infrastructure, that also often requires specific technologies, protocols, but also hardware requirements in terms of storage space, CPU utilization and RAM consumption.

For applications in IoT, systems, frameworks, mechanisms, and technologies that leave a small CPU/RAM footprint are preferred, but are also often an imperative. Blockchains typically requires larger processing capabilities than that of a typical node in an IoT ecosystem (e.g. smartwatch, embedded device, Raspberry Pi). For the integration between IoT and blockchains to be successful, Blockchain as a Service (BCaaS) should enable most of IoT devices to participate in the blockchain ecosystem, through tailored communication modules. Automation of existing tasks in blockchain deployment and operation presents a set of challenging tasks for BCaaS since blockchains represent highly distributed systems which are hard to deploy, manage, and operate on. Lastly, another relevant issue connected specifically to IoT devices is the lack of support for 64-bit operating systems (OS), as nodes in most blockchain frameworks require 64-bit OS.

BCaaS for IoT should allow multiple classes of heterogeneous devices to seamlessly communicate in a homogeneous blockchain environment. A BCaaS framework should be able to support integrating with heterogeneous environments – systems running on multiple physical machines with different technological footprints. That is why BCaaS aimed at IoT should have designated mechanisms to support deployment even to most resource-constrained devices (embedded devices, sensors, etc.). Creating and deploying a blockchain network is not a simple task – thus BCaaS should handle creating network infrastructure and later deployment in an easy, user-friendly manner through specific interfaces. Implementation of business logic should be enabled with tools and processes aimed at helping users, devices, or processes in generating and deploying the smart contract. Furthermore, efforts in BCaaS must be also directed towards monitoring - specifically blockchain exploration (transactions, blocks, users), device's health inspection (cpu/ram stats, tps, etc.) and topology changes capture, resource sharing and computational offloading and managing complex privacy models, but these concepts will not be the focus of this paper.

In this paper, we plan to address the above-mentioned challenges through showing that semi-automated blockchain deployment, business logic prototyping and integration with existing IoT systems can be easily managed through specifically tailored deployment workflows and mechanisms with Hyperledger Fabric (HF) blockchain technology. Specifically, our contributions rest in the following: Building an automated blockchain infrastructure specification and deployment component with Docker containerization (Hyperledger Fabric Infrastructure Configuration Generator - HFICG); Building an automated smart contract prototyping component with deployment mechanisms in-place (Hyperledger Fabric Chaincode Builder - HFCB); and Building a Hyperledger Fabric Operator Module (HFOM) to offer protected REST API-based communication with the underlying blockchain network (with a version of the module for resource-constrained devices). These components are implemented as parts of a commercial product [15].

The rest of the paper is structured as follows: In Section 2 we will discuss related work; in Section 3 we explain Hyperledger Fabric infrastructure and vital components; in Section 4 we elaborate on our proof-of-concept implementation; Section 5 offers a discussion over the solutions presented in Section 4; finally, we conclude the paper with Section 6.

2 Related Work

Integration of blockchain and IoT ecosystems has become an overarching industrial and academic topic covering many use-cases: IoT access control [8], secure firmware updates [5], energy systems [6], and so on. Concepts of collaboration frameworks for blockchain and IoT have been of research interest in terms of

digital asset management [11], management and configuration of IoT devices [4], secure information distribution [9], and so on.

Concerning HF, however, much of the actual research is focused on performance benchmarking models [14, 10]. While it is important to maximize the performance of the blockchain, enabling automated frameworks for generating configurations and deploying nodes is necessary, since HF brings a lot of complexity in configuration [17, 3]. Use cases like industrial IoT deployments (microcontrollers, sensors, etc.) benefit from automated processes since they rely on machine-to-machine communication with little to none human interaction.

Although there are an abundance of commercial BCaaS tools (i.e by Google, Amazon, Oracle, IBM and others SMBs [2, 16], etc.), none of them is equipped to tackle the challenges in using private blockchains in IoT systems through creating mechanisms for easy deployment, business-logic maintenance and high node churn resilience. To the best of our knowledge, this is a novel approach to addressing proof-of-concept tools, implementations, and discussion on the topic of autonomous onboarding of distributed systems to private permissioned blockchains, and automated business logic prototyping using Hyperledger Fabric, for IoT.

3 Hyperledger Fabric

Hyperledger Fabric is an enterprise-grade distributed ledger based on blockchain that leverages smart contracts to enforce trust between parties. It is a platform for distributed ledger solutions, underpinned by a modular architecture delivering high degrees of confidentiality, resiliency, flexibility and scalability [7].

Most important components of a HF blockchain are the domain, organizations, peers, orderers and certificate authorities. Domain represents a top-level network and project name. The organizations are the containers for the peers and respective certificate authorities (CA). Organizations are used for physical separation of the blockchain network to participating entities. The peers are nodes which are connected to clients and are responsible for committing transactions to the world state. Each peer has its copy of all transactions, and an organization can have multiple peers (i.e. for redundancy, physical separation). Ordering service provides a shared communication channel to clients and peers, offering a broadcast service for messages containing transactions. CA is responsible for creating HF certificates. It is used for verifying ownership in the network. Each certificate authority is tied to an organization. Other relevant concepts include channels and chaincode. Channels represent separate instances of ledgers and they are independent. A channel has a list of peers joined to it and instantiated chaincodes. Chaincode is the HF term for a smart contract. It represents the business logic behind a blockchain application written in programming code that can read from the ledger and write to the ledger. Chaincode is the only component that can manipulate the ledger. When the chaincode is created an endorsement policy must be provided. The policy can be any boolean logic (all peers, one peer from each organization, etc.) explaining who has to endorse a transaction. Policies are created per chaincode and are checked by the orderer service.

3.1 Addressing IoT BCaaS Requirements With Hyperledger Fabric

In our previous paper, we have identified 12 important BCaaS requirements for integration with IoT systems, categorizing them to 5 groups: Easy Network Specification and Deployment, Rapid Business Logic Prototyping, Resource Sharing, Complex Privacy Management Support, External Interoperability and

Data Federation [13]. This paper focuses on the first two categories and four requirements (see Figure 1), directly connecting these requirements with the Hyperledger Fabric DLT technology and highlight implementation directives for BCaaS. A proof-of-concept design and implementation for these requirements will be presented in Section 3, focusing on HFICG, HFCB, and HFOM. The tools addressed in this paper are implemented independently (and can be tested online) as part of a commercial product called ChainRider [15] (code not available publicly).

HF is suitable for BCaaS-IoT framework due to its in-built modularity of components, allowing them to be physically and logically distributed through the entire deployment ecosystem (IoT platform). The cost of running HF components (ordering service, peers, chaincodes) are rather small in comparison with Ethereum or Quorum. Specific tools can be built around automated HF infrastructure specification, generation, and deployment. Especially, leveraging HF binaries for cryptographic material generation can be automated, and complex networks can be built in a matter of seconds, provided that the tool has input about: blockchain network participants (organizations and peers), ordering service and certificate authority specifications, channels and chaincodes. Fast and easy writing and deployment of smart contracts can be enabled through specific software modules enabling input-based creation of Node.js, Golang or Java smart contracts with a basic interface and CRUD (and other) functions - similarly how you can inherit the basic token smart contract in Ethereum and build upon it. Deployment mechanisms, although not complex, can be automated by specifying channels and specific endorsement policies per chaincode.

IoT requirements addressed	Implementation directives	End objectives
[REQ1] Effortless specification of blockchain infrastructure for existing IoT systems	Create easy, user-friendly workflows to specify blockchain infrastructures	BCaaS has an infrastructure specification component for a specific blockchain network configuration
[REQ2] Effortless blockchain infrastructure deployment	Make direct and straightforward workflows for ease of deployment of a user-specified infrastructure	BCaaS has a specifically tailored step-by-step deployment mechanism that is well-documented
[REQ4] Fast writing of complex, enterprise business logic	Creating automated processes of writing smart contracts with basic functionalities (minimum CRUD operations for digital assets)	Automated rapid prototyping of business logic in IoT
[REQ6] Automated deployment of business logic	Create blockchain-specific mechanisms for smart contract deployment	Smart contract deployment is rather effortless and semi/fully automated

Fig. 1. Hyperledger Fabric BCaaS IoT requirements addressed in this paper

4 Proof-of-concept BCaaS Design And Implementation

This section covers implementation details of HFOM, HFICG, and HFCB. In this Section, we address **REQ1**, **REQ2**, **REQ4** and **REQ6** from our previous paper [13] (see Figure 1). **REQ1** and **REQ2** belong in the Easy Network Specification and Deployment category. Their combined aim is the effortless specification of HF blockchain infrastructure and automated deployment. **REQ4** and **REQ6** belong in the Rapid Business Logic Prototyping category, and their objective is to enable fast writing of complex enterprise business logic through automated workflows with instant-deployment enabled. The rest of the requirements will be addressed with implementation as part of our future work, and we do not provide implementation details nor discussion at this point. All of these requirements

address key IoT challenges: complex and distributed IoT topologies with heterogeneous devices having multiple technological footprints should have access to create and join a HF blockchain in a straightforward, guided workflow; IoT devices can create on-demand business logic components leading to rapid business functions modeling.

4.1 Hyperledger Fabric Operation Module

The Hyperledger Fabric Operation Module (HFOM) is a core component that handles secure communication between users, applications and the underlying blockchain infrastructure. The HFOM is a Node.js REST application that is designed to operate on any operating system with minimal CPU/RAM footprint. It is a dockerized service whose resources are accessible through a set of asynchronous token-based REST APIs. The design of the components and its services is completely asynchronous - as the underlying blockchain system naturally processes transactions asynchronously. To address the low resource requirements for IoT, HFOM offers three distinct deployment types:

1. Full HFOM with HF components running locally (on the same machine) - this HFOM requires HF peer node(s) to be deployed on the device running the HFOM. This deployment option offers fastest ledger Read/Write operations capabilities since it directly communicates with a locally running peer node. HFOM has all cryptographic materials of the network locally stored. It is implemented in Node.js. It is suitable for devices capable of addressing at least 1Gb of RAM memory and a 1.3 GHz CPU, such as the Raspberry Pi 3B. The capabilities also depend on the HF components running locally.
2. Full HFOM without HF components running locally - HFOM is connecting to a specific set of functioning peer nodes in the local IoT system. HFOM has all cryptographic materials of the network locally stored. It is implemented in Node.js. It is suitable for all devices with a minimum of 256Mb of RAM (such as Raspberry Pi 0, or Orange Pi 0).
3. Soft HFOM - connects to the closest full HFOM (1 or 2) and delivers its request using curl and HTTP or HTTPS. Its size is insignificant and it leaves the minimal CPU/RAM footprint and is suitable for most resource-constrained devices. It is implemented in Node.JS, Python, and Bash. It can run as a dockerized service, or as standalone service. It is suitable for devices that have under 128 Mb of RAM (such as Onion Omega 2).

To run properly the HFOM needs a blockchain network specification in *yaml* format. The specification is generated as part of the HFICG (Figure 2). HFOM on all levels can perform the following actions: enroll blockchain users, create channels, join peers to channels, install chaincodes, instantiate chaincodes, invoke chaincode functions, query ledger data, fetch specific blocks, fetch specific transactions, query installed/instantiated chaincodes on a specific channel and fetch channel information. HFOM on all levels have access to the HFICG and HFCB, as well as ledger data. Applications never directly interact with the core components (Peer, Orderer). All communication is proxied through the HFOM, building another security layer on top of already existing blockchain layer. Lastly, HFOM takes care of identity management on different levels of blockchain operation. By interacting with a Certificate Authority tied to the specific part of the network (usually organization), HFOM deals with certificate generation and revocation and transacting identities creation and removal on the blockchain. HFOM stores app certificates at any point in time until the access token tied to the certificate is expired, or the app's certificate has been revoked by the Certificate Authority. Access tokens are refreshed on a 10-minute basis, to minimize token hijacking probabilities.

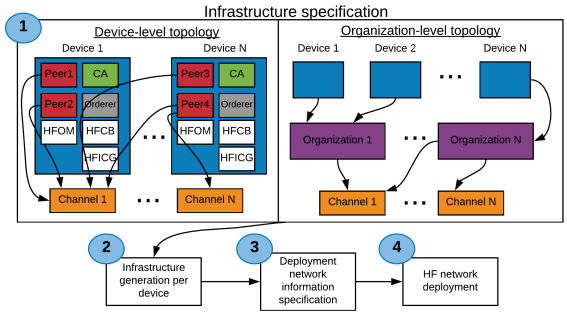

Fig. 2. Hyperledger Fabric Infrastructure Configuration Generator

4.2 Hyperledger Fabric Infrastructure Configuration Generator

HFICG is used for the creation of all necessary configuration and cryptographic material for an IoT system that is ready to join a HF blockchain network. HFICG component is designed to incorporate effortless creation of complex HF blockchain networks, with multiple organizations, channels, and various peer-channel connections, as well as provisioning all of these components to any desired number of physical machines. For any given system description that is provided to the HFICG, the component will generate a set of files and folders necessary for instant deployment and starting of the blockchain network, including HFOM deployment options of choice per physical machine. The workflow of HFICG is displayed in Figure 2.

Step 1 of Figure 2 indicates how infrastructure is specified in HFICG. The HF network infrastructure to be generated is described with two graphs, one being the device-level topology graph and the other organization-level topology graph. Through a simple user interface, these two graphs are made using drag-and-drop for adding HF and BCaaS components. In the same step, the graphs are transformed into an adequate JSON representation and passed to step 2. If an IoT system is to be a part of the Hyperledger IoT fabric network, a set of configuration files is required to be deployed together with the core Hyperledger Fabric components on each of the devices of an IoT system (step 2). All configuration and cryptographic material must be created on one machine, and later distributed to all parts of the system. The generated HF blockchain network deployment materials include: docker-compose file for the Ordering service, cryptographic material, genesis block, and a Bash startup script to bring up the Ordering service containers; docker-compose file (including: peers, state database and Certificate Authority - CA containers), cryptographic materials, channel creation and anchor peer configuration files, and a Bash startup script to bring up peers, state database and CA server configuration (including: channel creation commands, anchor peer update commands, channel join command per peer) per machine; HFOM/HFCB docker-compose files per machine and a Bash startup script to bring up the HFOM/HFCB services. To answer to the requirement of IoT systems having significant node churn, HFICG offers an easy extension of existing networks in terms of adding new organizations, devices, channels, and HF and BCaaS components, through the same mechanism displayed in Figure 2.

4.3 Hyperledger Fabric Chaincode Builder

Business logic in HF is specified through smart contracts or chaincodes in HF terminology. Chaincodes can be written in multiple programming languages: Golang, NodeJS, and Java. HFCB is designed as a fully-automated chaincode generation and deployment component. It handles all steps from describing the smart contact in a predefined format, to having the chaincode up and running on the blockchain network. This means that if there is an application interacting with the blockchain (case 1), and it is in need for a new smart contract that handles a new data stream, it can autonomously ask for a chaincode to be created through HFCB. Once the chaincode it is created by the HFCB the application is notified, and it can start interacting with the new chaincode. If there is a system user (human) interacting with the blockchain (case 2), this procedure is semi-automated with additional steps to cover on-the-fly smart contract editing. Both HFCB workflow cases are displayed on Figure 3.

Both workflow cases start with the description of the digital asset/record, central to the chaincode, available functions, and the programing language (step 1). A ledger record is usually specified as an arbitrary Javascript (JSON) object. Essentially, when generating a smart contract, we are specifying two things: 1) how this record object will look like (its attributes and their types) and 2) what functions are going to be performed with it. For illustration, let us consider a simple IoT use case: we have a smart home environment with temperature sensors collecting data from different rooms in the house. To record these sensory data on the HF blockchain we need to have a chaincode that will represent temperature reading objects and allow for sensory data to be written on the blockchain. Every temperature reading that is going to be written to the blockchain will contain the room where it was recorded, the value that was recorded, and a unique key. Room will be a string such as a kitchen or living room, value is the temperature value. HFCB offers to include following functions on the given asset: insert asset, update asset, update asset by attribute value, associate asset with a different key, remove asset from the state, query asset by key, query asset by key range, query asset by CouchDB query string. An application can use a subset of the functions.

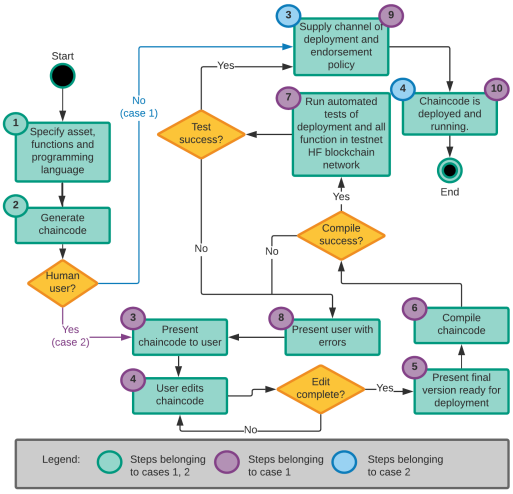

Fig. 3. Hyperledger Fabric Chaincode Builder

Programing languages of choice are Node.js, Golang, and Java. Both workflows continue to the generation of code and dependency files (step 2).

Once the chaincode files have been generated, the application/device is asked to supply the deployment channel and the endorsement policy (step 3, case 2). After that (step 4, case 2) the chaincode is up and running, and ready to serve requests.

The case 1 workflow continues by presenting the generated chaincode to the user (step 3, case 1). On the chaincode, a user can perform many edits (step 4, case 1) to add new functions, change existing ones, etc. When editing is complete user is presented with the final version (step 5, case 1). This final version is then compiled (step 6, case 1) and if the compilation succeeds the chaincode is deployed and tested in a testnet blockchain (step 7, case 1). If the compilation fails, the error messages are presented to the user (step 8, case 1), and the user is back at step 3 of case 1. Outputs from step 7 of case 1 are presented to the user, and if not satisfied, the user can go back to the editing step (step 4, case 1). While step 6 eliminates all possible syntax errors, step 7 makes sure that there are no chaincode runtime errors. If deployment tests finished without errors, the user is asked to select a channel where the chaincode will be deployed and the endorsement policy (step 9, case 1). Once that step is completed the chaincode is up and running and begins accepting transactions (step 10, case 1).

5 Discussion

The implemented components are used and tested in three real-world IoT deployments: Agile IoT platform (see Acknowledgement Section), the Arizona State University Blockchain Research Laboratory Carbon trading IoT platform [1] and our Asset Tracking fog computing solution [12]. During development and testing, we have generated 50 HF infrastructure configurations with varying complexity, and 150 chaincodes for deployed IoT solutions mentioned in the previous sentence.

The most complex example we have tested with HFICG is the following infrastructure specification: 30 organisations with 40 channels (1 channels per organisation, and 10 channels shared between multiple organisations); Kafka Ordering service (7 Kafka nodes, 5 Zookeeper nodes); a CouchDB state database Docker container per machine (omitted for low machine CPU/RAM capabilities); a HFOM per machine (with the suitable deployment option (see Section 4.3) depending on the machine CPU/RAM capabilities); a Certificate Authority container per organisation; Each organisation runs 5 machines, making a total of 150 machines for this deployment; Each machine runs 0-2 endorsing peers (depending on the machine CPU/RAM capabilities), making a total of 300 peers. This HF configuration is generated with HFICG within 7 minutes. With network information supplied for all participating machines, we have measured the amount of time necessary to: 1. deploy configuration files; 2. start HF components in Docker containers; 3. Perform channel creation, peer channel joins, and anchor peer updates. For example, on a Raspberry Pi 3 B+ device, steps (1), (2) and (3) are done within 3 minutes. Note that this is the time in an ideal scenario. Due to network issues and deployment retry-policies, the deployment time varies.

We have simulated the above-mentioned deployment on the Azure Cloud Platform. Virtual machine specifications regarding CPU and RAM were set to lowest to mimic typical IoT devices – two machines types were randomly assigned to each device: first having 1 CPU and 0.5 Gb RAM, second having 1 CPU and 1 Gb of RAM. The machines with lower settings run 0-1 peers (assigned randomly), HFOM with deployment option 3 (see Section 4.1), and no CouchDB

container. The Ordering service node was deployed to a stronger machine (8 CPUs, 32 Gb RAM). Since deployment is done in parallel for all machines, the entire blockchain network was deployed, set up and running in under 15 minutes. Note, that most of the time is spent on machines downloading necessary Docker images to run HF components and setting up HFOM (approx. 70%).

The most complex example we have tested with HFCB is a chaincode for a digital asset with 30 attributes, having all functions (CRUD and three different query functions) with a CouchDB index created for 15 attributes when performing inserts. The chaincode is generated within 3 seconds, and deployed (through a HFOM setup on a Raspberry Pi 3 B+), up and running within another 100 seconds, making a total execution time under 2 minutes.

In conclusion, one can have a complex, fully functional HF network with on-demand business logic up and running in under 25 minutes with HFICG, HFCB and HFOM. Furthermore, with HFICG and HFCB the time required to deploy infrastructure and business logic is significantly diminished over manual writing of configuration files, moving files to dedicated machines and starting all HF components.

Lastly, we argue that security and privacy of IoT systems can be strengthened through using a blockchain infrastructure such as HF. Transparency levels for data can be adjusted according to the desired use case through channels modeling or private data specification. Since channels represent physically separate data ledgers, data privacy and security can be managed within a certain set of organisations, even peers. User/Device level attributes can be tied to user/device certificates and access policies for data can be enforced through chaincodes at runtime. To ensure proper transaction validation, chaincode endorsement policies can be set to have varying endorsement requirements. Data access is secured via strict checking of signatures/certificates at all steps: HFOM/HFICG/HFCB, peer level and chaincode level. Unauthenticated reads and writes are thus automatically discarded. Having a supportive HF infrastructure makes on-demand traceability and auditability of actions simpler. Malicious actors and actions can be detected, and machines/users blacklisted.

6 Conclusion

In this paper we have built a proof-of-concept BCaaS system using Hyperledger Fabric blockchain technology. This paper showcases implementation details from a set of BCaaS requirements for IoT systems defined in our previous paper on the topic [13].

We have discussed BCaaS in IoT, and provided an implementation of several functionalities, making this PoC the first, relevant BCaaS PoC that considers: automated blockchain network infrastructure generation and deployment in a physically distributed system environment and rapid business logic prototyping through hastened smart contract writing and deployment. With HFICG configuration specification, infrastructure setup and deployment time are significantly diminished. HFCB makes a large impact where fully-automated on-demand chaincode generation and deployment is necessary. Lastly, HFOM makes a move towards providing a tool for fast and secure communication with the underlying blockchain infrastructure. This framework of tools has been already validated through integration with Agile IoT platform. Furthermore, our framework has been used by Arizona State University Blockchain Research Laboratory in context of their Carbon trading IoT platform development and is providing a HF blockchain network for our BLEMAT Asset Tracking IoT solution. The tools mentioned in the paper are offered as parts of a commercial solution [15].

As part of the future work our focus will be complementing the BCaaS framework until the complete set of BCaaS services from our previous paper is

implemented. We will aim for integration with other IoT platforms, while also carefully reviewing new BCaaS requirements, implementation recommendations and end objectives. Lastly, IoT platforms are always subject to a degree of uncertainties, particularly as regards as their number of components and the frequency that they are running. As part of future work we will work on larger experimental studies to see how the BCaaS could work in an environment when new components are frequently added or removed over time.

Acknowledgement

The research and outcomes behind this paper are a part of the open call research project for Agile IoT, H2020-EU.2.1.1. - INDUSTRIAL LEADERSHIP - Project ID: 688088. The authors also thank the Ministry of Education, Science and Technological Development of the Republic of Serbia for support through project no. OI174023 – "*Intelligent techniques and their integration into wide-spectrum decision support*".

References

1. Arizona State University Blockchain Research Laboratory: Carbon trading on a blockchain (2018), https://blockchain.asu.edu/carbon-trading-on-a-blockchain, Last accessed: 18-05-2019
2. ChainStack: Managed blockchain infrastructure. https://chainstack.com/, accessed: 2019-08-10
3. Gaur, N., Desrosiers, L., Ramakrishna, V., Novotny, P., Baset, S.A., O'Dowd, A.: Hands-On Blockchain with Hyperledger: Building decentralized applications with Hyperledger Fabric and Composer. Packt Publishing Ltd (2018)
4. Huh, S., Cho, S., Kim, S.: Managing iot devices using blockchain platform. In: 2017 19th international conference on advanced communication technology (ICACT). pp. 464–467. IEEE (2017)
5. Lee, B., Lee, J.H.: Blockchain-based secure firmware update for embedded devices in an internet of things environment. The Journal of Supercomputing **73**(3), 1152–1167 (2017)
6. Li, Z., Kang, J., Yu, R., Ye, D., Deng, Q., Zhang, Y.: Consortium blockchain for secure energy trading in industrial internet of things. IEEE transactions on industrial informatics **14**(8), 3690–3700 (2018)
7. Linux Foundation: Hyperledger fabric blockchain (2019), https://hyperledger-fabric.readthedocs.io, Last accessed: 18-05-2019
8. Ouaddah, A., Abou Elkalam, A., Ait Ouahman, A.: Fairaccess: a new blockchain-based access control framework for the internet of things. Security and Communication Networks **9**(18), 5943–5964 (2016)
9. Polyzos, G.C., Fotiou, N.: Blockchain-assisted information distribution for the internet of things. In: 2017 IEEE International Conference on Information Reuse and Integration (IRI). pp. 75–78. IEEE (2017)
10. Pongnumkul, S., Siripanpornchana, C., Thajchayapong, S.: Performance analysis of private blockchain platforms in varying workloads. In: 2017 26th International Conference on Computer Communication and Networks (ICCCN). pp. 1–6. IEEE (2017)
11. Samaniego, M., Deters, R.: Blockchain as a service for iot. In: 2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData). pp. 433–436. IEEE (2016)
12. Saša Pešić, e.a.: Blemat-context modeling and machine learning for indoor positioning systems (2019), submitted for review in International Journal on Artificial Intelligence Tools (ISSN: 0218-2130)
13. Saša Pešić, e.a.: Conceptualizing a collaboration framework between blockchain technology and the internet of things (2019), accepted to CompSysTech'19

14. Thakkar, P., Nathan, S., Viswanathan, B.: Performance benchmarking and optimizing hyperledger fabric blockchain platform. In: 2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS). pp. 264–276. IEEE (2018)
15. VizLore: Chainrider blockchain as a service. https://chainrider.io/, accessed: 2019-08-10
16. Worldsibu: Worldsibu: The enterprise blockchain development platform. https://worldsibu.tech/, accessed: 2019-08-10
17. Yewale, A.: Study of Blockchain-as-a-Service Systems with a Case Study of Hyperledger Fabric Implementation on Kubernetes. Ph.D. thesis, University of Nevada, Las Vegas (2018)