Blockchain Performance Analysis for Supporting Cross-Border E-Government Services

Dimitris Geneiatakis, Yannis Soupionis, Gary Steri, Ioannis Kounelis, Ricardo Neisse, and Igor Nai-Fovino

***Abstract*—With the continuous development of distributed ledger and blockchain technologies, new use cases apart from cryptocurrencies have come into the spotlight. In this article, we evaluate whether an e-government service could be a suitable candidate for a blockchain transformation. We selected as a reference test system an existing cross-border e-government service that is used for supporting goods exchanges across the European Union. We show how such an indicative paradigm can be transformed into a blockchain system. In order to do so, we deployed it in an emulated architecture for evaluating its performance under various realistic conditions. Our results show that the deployed system is able to meet the requirements, both in terms of throughput and transaction speed. Moreover, it shows clear advantages in terms of usability and synchronization between all entities.**

***Index Terms*—Blockchain, e-government, Hyperledger Fabric, performance evaluation.**

I. Introduction

THE diffusion of distributed ledger and blockchain technologies is strictly linked to the digital world and cryptocurrencies. Undoubtedly, their popularity has enormously grown due to systems such as Bitcoin and Ethereum, which have highly attracted people attention mainly as alternative payment and trading platforms. The reason behind their success is mainly due to the possibility of deploying decentralized architectures where parties can transact with each other without the need of a centralized trusted entity; indeed they rely on peer-to-peer networking architectures. Moreover, the intrinsic properties of blockchain ensure data immutability, provenance and transparency for the accomplished transactions.

However, cryptocurrencies are not the only area where distributed ledgers and blockchains can be applied. During recent years, several other use cases have been proposed in different domains such as food supply chain [1], [2] or goods tracking [3]. An important area where these technologies could bring their advantages is certainly represented by the *electronic government (e-government)* services, which does not include only interactions with citizens, but also between institutions [4] known also as government to government (G2G) services (including cross boundaries interactions, i.e., between countries/states).

In e-government services, aspects related to transparency and transactions security play a fundamental role for establishing trust in the provided services. Current approaches, especially for G2G services, establish security using either private communication channels or well-known security communication protocols over public networks. Both approaches are valid and functional, however, on one side they do not guarantee data security by design, i.e., integrity for the whole data life cycle, and might raise security issues as reported in various research works [4]–[6]. On the other side, they may not support nonrepudiation and liability services. Furthermore, current e-government services face data synchronization issues because administrative and legal requirements should be fulfilled first. Also currently most of the e-government services rely on a centralized service, a single point of failure.

Besides, it should be noted that in e-government services trust issues, in general, can also be raised especially when dealing with stored data, transactions validity, service and systems' conformity among "distributed" entities. These aspects growing in importance when cross-boundary integration is required, for instance when considering different countries, state collaboration for achieving efficient intergovernment service functionality. A detailed analysis of trust issues in e-government services is out of the scope of this article and can be found in [7].

Such challenges in e-government services can be solved by using blockchain infrastructures as they are secure by design and data sharing can be triggered when specific criteria are satisfied through the use of smart contracts in completely transparent and automated way. Indeed, blockchain technologies offer a powerful framework for decentralized data processing and sharing, *but how suitable are they for e-government services? Will scalability be a problem? Which are the requirements in terms of hardware infrastructure?* Such questions can be raised whenever an e-government service is proposed to be converted into a *blockchain government(b-government)* service.

This article provides an analysis to support policy makers in deciding whether the adoption of blockchain overcurrent infrastructures and resources can efficiently support b-government services. In this context, the Directorate-General for Taxation and Customs Union of the European Commission is exploring

Manuscript received June 30, 2019; revised November 15, 2019 and January 21, 2020; accepted February 28, 2020. Date of publication June 30, 2019; date of current version October 9, 2020. This work was supported by the Directorate-General for Taxation and Customs Union of the European Commission for exploring the possibility of using blockchain technologies for data-sharing scenarios, with a view on enabling more sophisticated use cases in which trusted data play a key role. Review of this manuscript was arranged by Department Editor K.-K. R. Choo. *(Corresponding author: Dimitris Geneiatakis.)*

The authors are with the Cyber and Digital Citizens' Security Unit, European Commission—Joint Research Centre, 21027 Ispra, Italy (e-mail: dimitrios.geneiatakis@ec.europa.eu; yannis.soupionis@ec.europa.eu; gary.steri@ec.europa.eu; ioannis.kounelis@ec.europa.eu; ricardo.neisse@ec.europa.eu; igor-nai.fovino@ec.europa.eu).

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TEM.2020.2979325

the possibility of using blockchain technologies for data-sharing scenarios for supporting G2G services among European Union (EU) Member States Authorities (MSAs).

We opt for an existing e-government service that facilitates the excise goods monitoring system between MSAs. We convert it into a *b-government* service to explore its potentiality, operability, and performance. An experimental version of the service is deployed over the Hyperledger Fabric, which is a permissioned blockchain platform, on a modularized environment that is capable of emulating real network conditions and architectures depending on the needs. The configured blockchain system is made of 28 nodes that represent MSAs in the service. We evaluate the b-government service under different network conditions, i.e., number of transactions sent to the service, network latency, and bandwidth, and we analyze how the blockchain network size can influence system performance.

The results of our article show that on one hand transforming a data-sharing e-government to b-government service is technically feasible taking into consideration specific network configurations. For instance, in a network configuration with 4 Mb/s and latency 3 ms client transaction round-trip time approximates to 1 s, and the whole network can efficiently support throughput up to 8 transactions per second (TPS), whereas if the available bandwidth is 1 Gb/s the maximum throughput can reach up to 48 TPS. On the other hand, system management could be simplified, as the use of a smart contract ensures that the same application logic is used and shared to all the participants, and the intrinsic blockchain characteristics guarantee fundamental security requirements by design. To the best of our knowledge, this is the first article that evaluates at scale a b-government service, i.e., considering 28 participant nodes, over a permissioned blockchain platform and realistic networking configurations.

The rest of this article is organized as follows. Section II provides a background on blockchain technologies and highlights their main characteristics. Section III describes an e-government service as an indicative example suitable for blockchain transformation. Section IV determines a blockchain platform, among different options, for implementing such a service. Section V analyzes the experimental architecture in which the service is deployed for emulating real network and system conditions. Section VI reports on the service performance considering different configurations. Section VII overviews the related works in blockchain with emphasis on e-government applications, and performance evaluations. Finally, Section VIII concludes this article.

II. Background on Blockchain

Distributed data storage system is a well-studied research domain that facilitates data storage, sharing, and synchronization in different nodes over a network. In fact, the very first distributed database system was introduced in 1980 [8]. In this context, until now different distributed solutions such as Dynamo [9] and Cassandra [10] have been proposed to fulfill different needs and requirements.

With the advent of distributed ledger technologies (DLTs) [11] and blockchain systems, data consistency across nodes, integrity, and immutability can be achieved by design. The term

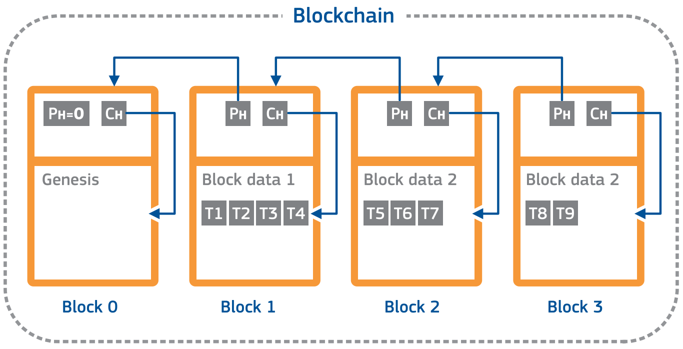

Fig. 1. Blockchain data structure high-level representation. Each block holds users' data, a hash of the current block (C_h), and a link to the previous block (P_h).

blockchain actually identifies the underlying data structure of the first types of DLTs, even though nowadays not all the DLTs are based on a blockchain. The blockchain records and synchronizes data in blocks of transactions chained with the support of cryptographic techniques. Each block contains the cryptographic hash of the previous block, a time stamp and the transaction data. In this way every block is cryptographically linked to the previous one, making the list of records immutable. In fact, changing data in the chain would imply the recalculation of the hashes, unfeasible from an organizational and computational point of view. Fig. 1 illustrates an abstraction of a blockchain data structure.

In a typical blockchain scenario, end users submit transactions to a peer-to-peer network where particular type of nodes validate and insert them in a new block, which is then propagated to all the participants. Each transaction is digitally signed by the user with its private key, so no other entity can claim the authorship neither can the user repudiate it.

All the participants in a blockchain network hold their own copy of the data, and can thus calculate independently the current known "state" of the system. As a result, there is no single point of failure, in contrast to centralized data storage services. Due to a synchronization mechanism that the network supports in case of failure of a participant, the latest state of the system can be resumed, so that all the participants, at any time, share a common ground truth. Indeed, this is achieved using a Byzantine fault tolerance (BFT) [12] consensus mechanism such as proof of work [13] and proof of stake (PoS) [14]; however, a detailed analysis of consensus mechanisms is out of the scope of this article. We nonetheless refer the interested reader to relevant state-of-the-art related works [15], [16].

Another important characteristic of blockchain systems is the way users access them in order to read, submit, and validate transactions. When anyone can read and access the system, the blockchain is categorized as public; anyone can fetch the whole blockchain and read its contents. The opposite is to have a private blockchain, where only authorized entities have access. Similarly, depending on whom can submit and validate transactions, the blockchain is called permissionless or permissioned.

With the development of Ethereum,[1] blockchain systems introduced the notion of *smart contract* functionality. A smart

[1][Online]. Available: https://ethereum.org/

contract is a computer program that is capable of executing or enforcing a predefined agreement using a blockchain, when and if specific conditions are met. It is deterministic[2] and benefits from the intrinsic properties of a blockchain system in order to deploy a service that even "trustless" entities could trust.

III. Identified Scenario

This section provides an overview of an existing e-government service that requires data sharing among different entities. The service that we have identified is in the context of monitoring the movement of excise goods, such as tobacco and alcohol, across EU Member States (MS) and is called Excise Movement and Control System (EMCS). In particular, we are interested in the facilitator of the data-sharing platform that EMCS is using, the System for Exchange of Excise Data (SEED).

A. *System for Exchange of Excise Data*

Common rules for exchanging excise goods across MS are governed by the Directive 2008/118/EC.[3] Excise goods that are transferred within the EU territory have to be authorized and monitored by the appropriate MSAs, during the whole life cycle of the exchange procedure. Information about the goods and their movements is recorded in an electronic administrative document (eAD), which is the core of EMCS, and its structure is described in Commission Regulation (EC) No 684/2009.[4] eADs are submitted by a consignor to the MSA of dispatch, which validates and forwards them to the final destination (consignee) via the corresponding MSA.

In order to allow a prompt and correct validation of the eADs, both consignors and consignees (merchants) must be registered and authorized by the respective MSA. Such registration records that are called trader authorizations consist of the following:

1) merchant identification number;
2) merchants name and address;
3) types of goods;
4) identification of the liaison office;
5) date of registration;
6) record ending date;
7) registration offices.

MSAs are responsible not only to validate and confirm the correctness of merchants records but also to check their compliance with SEED rules before distributing them to other MSAs through SEED. Indeed, SEED is a repository that, among the others, maintains and distributes all the relevant information regarding merchants among MSAs.

More specifically, whenever a new merchant registration, or an update of an existing record occurs, the corresponding MSA validates its correctness, and "pushes" the new data into the local service, i.e., the data storage of an MSA. Afterward, the updated records are communicated to the central service that performs conformity checks to control whether the data comply

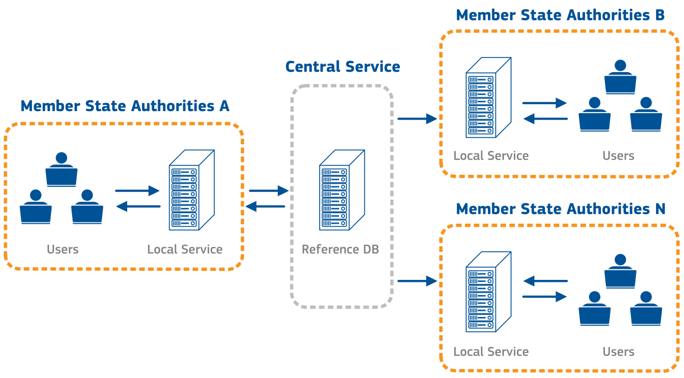

Fig. 2. SEED architecture data registering overview. Users at MSAs introduce new data and push them in the central service, which is responsible to distribute to all the relevant participants in order to keep their databases updated.

with the underlying rules. If this is the case, the relevant MSA is informed, the registration data are integrated in the central service (i.e., reference DB), and distributed to all MSAs. Here, it should be noted that according to the data synchronization rules of SEED, the following conditions hold:

1) the central service should make available the new data to all the MSAs at the latest on the day when they become legally valid;
2) each MSA is responsible for updating its national registration repository as soon as they receive any updates from the central service.

The whole procedure for registering and distributing SEED information is illustrated in Fig. 2. A detailed analysis of the SEED can be found in [17].

B. *Why SEED on Blockchain?*

In this section, we overview the benefits and challenges for transforming "SEED like" e-government services into b-government services. Currently in SEED, MSAs are responsible for integrating their local service with the central one in order to share and keep merchants' related information updated. This means that MSAs have to develop their own software, as a local service, that is capable of communicating with the central service and fulfilling the SEED functionality and requirements. On one side, this gives to an MSA more freedom and flexibility on the technical implementation. On the other side, such an option can have the following drawbacks.

1) Compatibility issues could arise in case the requirements for local service implementation were not respected.
2) Updating the functionalities is more complex since all different implementations should be separately updated.
3) The overall maintenance cost can be high.

Furthermore, all the data-sharing communications in SEED are made asynchronously in a time frame between 15 min and 2 h, whereas as already described in Section III-A each MSA has to update its national registration repository as soon as it receives updates from the central service. Therefore, data are not shared at real (or near real) time among MSAs. In addition, the current implementation relies on a separate infrastructure in order to ensure security. Though this approach is valid and functional, it does not by design guarantee data integrity, authenticity, and identity

[2]Smart contract's output is the same for everyone who executes it.

[3][Online]. Available: https://eur-lex.europa.eu/legal-content/GA/TXT/?uri=CELEX:32008L0118

[4][Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32009R0684

of the originator. Finally, the architecture of a Trans-European System, such as SEED (see Fig. 2), relies on a central service for the dissemination of information to all the participants. As the central service poses a single point of failure it has strong requirements in terms of reliability and performance.

We are interested in investigating blockchain potentiality on data-sharing services such as SEED. This is because blockchain is *secure by design* in certain aspects, i.e., preserving data integrity, and enabling data sharing in near real time among entities that do not necessarily need to trust each other when specific requirements are met. With the use of smart contracts all entities would share the same solution and run a common logic, ensuring at any time, consistency of the data validation procedures, and reducing the total maintenance cost, compared to a situation where each entity maintains its own solution. This way the global trust could be enhanced.

Moreover, using a blockchain architecture for such a system, would guarantee high reliability and performance of the global system without dependence on a high-performance central service, thus making the system more resilient. As a result, data sharing could is provided in a secure, immutable, traceable and transparent way.

At this point, one might argue that 1) data changes might be challenging because such functionalities are not directly supported, nevertheless they can be enforced at a data management layer as all the data history is recorded (in the raw blockchain data), and 2) in such a federated application environment similar approaches like [9], [10] can be used. However, they lack built-in security mechanisms, and do not support common logic sharing over a data distributed mechanism among all participants.

IV. SEED on Blockchain Implementation

In order to implement SEED on blockchain, we need to first identify the most suitable platform. To do so, we set the requirements that are needed in such a service, and determine which platform better meets them, from a technical and architectural point of view. Afterward, to assess the performance of SEED in blockchain we relied on an environment that is capable of emulating real network architecture. On top of the emulating architecture, we built a b-government service (for SEED) using Hyperledger Fabric (ver. 1.1) and considering different network parameters to identify possible system bottlenecks and resources restrictions. Our performance evaluation relies on well-known metrics such as request round-trip time and utilization of system resources. All the details of our approach are reported in the following sections.

A. Blockchain Platform Options

The technical choice of implementing any e-government service on a blockchain relies on a variety of factors. Specifically, in SEED like e-government paradigms the following requirements must be covered.

1) *Req. 1*: Service data cannot be public due to legal obligations.
2) *Req. 2*: Common logic should be shared among all the participants.
3) *Req. 3*: Service data should be shared as soon as administrative and legal rules are met.
4) *Req. 4*: Cost should be as low as possible.

Currently, various types of blockchain platforms have been developed, and can be used for deploying related services. Table I overviews the most well-known platforms considering their different characteristics, e.g., smart contract, consensus mechanisms, etc., as they have already been described in Section II.

Based on the available options those that provide public access to data are excluded, such as Bitcoin, Ethereum, IOTA,[5] and Ripple, due to data-sharing restrictions (*Req. 1*). Even if these systems can be deployed separately from the main network (e.g., Ethereum privatenet), they still do not provide access control mechanisms that would exclude other entities from joining the network, when using a public IP address.

Furthermore, as all the entities must share the same logic, smart contract functionality should be supported (*Reqs. 2 and 3*). As a result, the available options are the Hyperledger family solutions and EOS.[6] However, since public administration would like to support services at the lower possible cost, platforms that introduce transaction fees are excluded as well (Req. 4). Therefore, Hyperledger Fabric and Sawtooth are platforms without fees that support consensus mechanisms requiring the collaboration between "participants," which is a suitable model for e-government services.

Between the two, we chose Hyperledger Fabric as the consensus mechanism is more suitable for our specific experiment. Sawtooth uses a consensus mechanism that assigns to a random participant the validation of a transaction, i.e., proof of elapsed time. However, in SEED like services we want to define *a priori* who the validator will be, and in particular who is entitled to endorse specific operations.

B. SEED Flow on Hyperledger Fabric

In a typical SEED service scenario, users at *local* services, i.e., National Administrations, submit their transactions, which afterward are forwarded to the ordering service in order to disseminate the corresponding data to all the MSAs (see Fig. 2). At this point, it should be noted that all the participants in the network are uniquely identified through digital certificates, which are embedded in the exchanged messages. To achieve the data-sharing service over the Hyperledger Fabric infrastructure, the transaction flow will be the following.

1) *Transaction submission*: A client submits a signed transaction proposal to the corresponding MSA node, in which it includes also his/her identity (X.509 certificate).
2) *Transaction verification*: The MSA node verifies the validity of the transaction, simulates the execution of the related smart contract functionality, and returns a signed endorsement result to the client. The identity of the MSA node (X.509 certificate) is also included in this response.
3) *Transaction ordering*: The client sends the transaction proposal and the related result to the ordering service. The

[5][Online]. Available: https://www.iota.org/

[6][Online]. Available: https://eos.io/

TABLE I
BLOCKCHAIN PLATFORMS

Solution	Read Access	Send & Validating Tran.	Consensus	Smart Contract	Fee
Bitcoin	Public	Permissionless	Proof of Work	No	Yes
Hyperledger Fabric	Private	Permissioned	Modular (PBFT, CFT)	Yes	No
Hyperledger Sawtooth	Private	Permissioned	Proof of Elapsed Time	Yes	No
Ripple	Public	Permissionless	Ripple	No	Yes
Ethereum	Public	Permissionless	Proof of Work	Yes	Yes
IOTA	Public	Permissionless	Proof of Work	No	No
EOS	Private	Permissioned	Delegated Proof of Stake	Yes	Yes

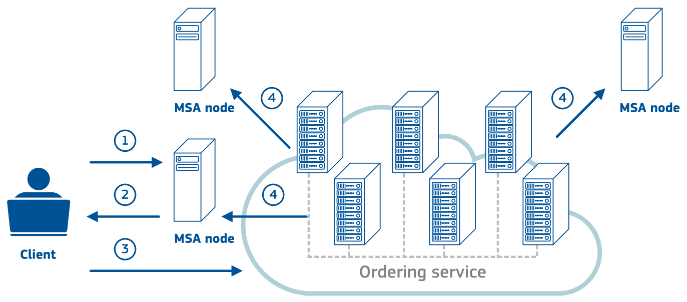

Fig. 3. Overview of transaction flow on Hyperledger Fabric.

latter orders all the transactions coming from different clients, builds the blocks and broadcasts them to all the MSA nodes.

4) *Transaction commitment*: MSA nodes receive the blocks of transactions from the ordering service and, upon verification of their validity in the context of the deployed policy, e.g., at least one MSA node should have signed the transaction, write them in their local copy of the ledger.

All the underlying network communications can be (optionally) protected by transport layer security (TLS) [18]. The above-mentioned procedure is illustrated in Fig. 3, whereas the details for the MSA nodes and the ordering related services are specifically described in Section V-C. A detailed analysis of the Hyperledger Fabric architecture can be found in [19].

V. Experimental Infrastructure

In this section, we briefly present our experimental infrastructure and the steps followed to recreate the emulated information and communications technology (ICT) network topology, which accommodates our SEED like service in order to assess the general performance of the system.

A. Experimental Platform for Internet Contingencies (EPIC) Overview

The EPIC [20] is a hybrid facility for studying the security and stability of distributed systems. The architecture of EPIC relies on an emulation testbed based on the Deter software [21], [22] in order to recreate the cyber part of a distributed system, e.g., servers and corporate network.

By employing an emulation-based testbed we ensure strong fidelity, repeatability, measurement accuracy and safety of the cyber layer. This approach is well-established in the field of cyber security [23]–[25] and was chosen in order to overcome the major difficulties that arise while trying to simulate the behavior of ICT components under stress, attacks, or failures. In terms of resources EPIC consist of 356 experimental nodes, eight switches only for the experimentation infrastructure and a few special physical equipment, such as programmable logical controllers.

B. Recreating Cyber Systems

The Deter adaptation for testing purposes of advanced network systems allows the assignment of computational resources to a virtual network topology. The EPIC can set parameters to the physical infrastructures in order to implement and emulate a realistic network topology. The process is made as transparently as possible. This way provides significant advantages in terms of repeatability, scalability, and controllability of our experiments.

The Deter software comes with a Web interface where it is possible to assign physical equipment in order to create emulated networks. Briefly, the main steps of the process are as follows.

1) Write the experimental script, which describes in detail the network architecture/topology. In order to compose the script, the network simulator 2 (ns2) scripting language is used. This eases the process for future reuse of our experiment.
2) The EPIC software then initiates the experiment. Based on the script, it reserves and assigns the necessary physical resources. The process of allocating resources to someone's experiment is called swap-in.
3) Additionally, EPIC creates virtual private networks on the network switches in order to construct the network topology by connecting experimental nodes. Then, the software adjusts other parameters of the network, such as latency and packet loss, by assigning additional nodes for these specific purposes.
4) Applications and services specific for the experiment, e.g., Docker containers, can be initiated and configured either automatically or manually, by logging in to each node that has been assigned into the specific experiment.

C. SEED Blockchain Service Configuration on EPIC

Our experimental testbed blockchain architecture for a SEED like service relies on the Hyperledger Fabric. From an

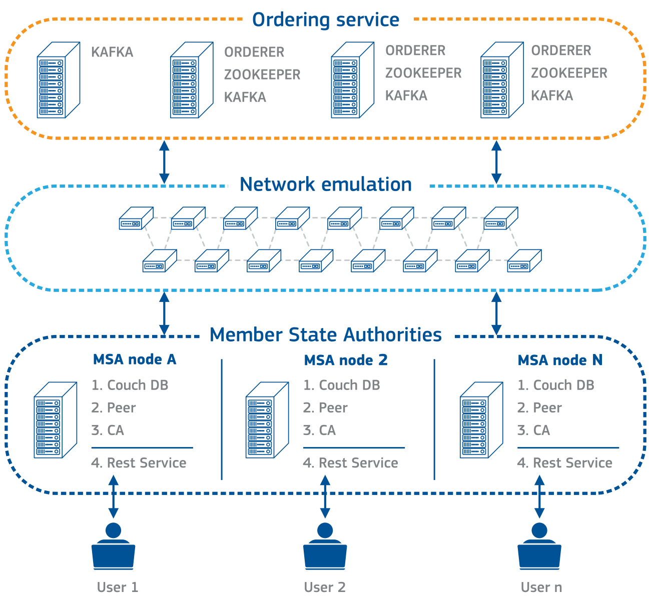

Fig. 4. SEED Hyperledger Configuration on EPIC infrastructure.

architectural point of view the system consists of the "ordering" service and the "distributed" components, i.e., MSA nodes. The high-level architecture is illustrated in Fig. 4.

The ordering service is managed by "trusted" authorities among the participants, meaning that it can be distributed in different locations, and not all the components need to be physically colocated; this way there is no single point of failure. In fact, any trusted entity in the network can potentially operate an ordering service node; however, this should be defined during blockchain network bootstrapping procedure.

The main task of the ordering service is to sort the messages/requests exchanged between the participants. It consists of the following services: ZooKeeper (three instances), Kafka (four instances), and Orderer (three instances). Each of the instances of a specific service is executed on a different machine, and thus can be distributed onto different places. This is the minimum configuration in terms of capacity for supporting failover at the ordering service. The number of the required instances for each of the supporting services is defined in the Hyperledger Fabric specifications.[7] In particular, in this setup, only one instance for each of the different services is "allowed" to be in fail status without affecting the ordering service availability. For instance, if one of the Kafka instances is not responding, the ordering service is still functional; however, if a second one becomes unavailable, then the ordering service will not be functional.

The distributed components are managed by the participants, i.e., MSAs in the network and require management only at "local" level. All the services for each MSA are hosted on a single machine. MSAs nodes main goal is to endorse the transactions proposed by the clients and receive the ordered blocks of transaction from the ordering service to maintain their local copy of the ledger. In particular, each MSA operates the following services.

[7] [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-1.4/orderer/ordering_servi ce.html

1) *CouchDB*, the database that maintains the world state (all the valid transactions) of the blockchain and allows the storage of javascript object notation (JSON) objects.
2) *Peer*, a core service in the Hyperledger Fabric architecture as it stores the ledger and validates the transactions according to the defined policy.
3) A *certificate authority* (CA) that provides digital identities (certificates) to the participants (users) of the local organization to the blockchain network supporting a complete life cycle of the generated certificates.
4) A *smart contract* that implements basic SEED specification checks such as user access control and message conformity.
5) An *application interface* for interacting with the blockchain, which is implemented as a representational state transfer (REST) service. This component accomplishes all the interactions (as described in Section IV-B) on behalf of the user in order to commit a transaction in the blockchain network. In fact, this component generates the trader authorization requests in JSON format.

At this point, it should be noted that all the Hyperledger Fabric related services both for the ordering service and MSA peer nodes subsystems are configured and executed using the corresponding docker images with the standard deployment option. Moreover, in our configuration all the underlying network communications, between the participants (clients, peers, and the ordering service), are securely protected by TLS. All the required certificates and private keys both for TLS and for the blockchain services are generated during blockchain network initialization procedure according to Hyperledger Fabric specifications.

VI. Performance Evaluation

In this section, we study the performance of our system in terms of computational resources and its responsiveness. To do so, we use as indicators memory, central processing unit (CPU), and end-to-end service delivery under various testing scenarios on the EPIC infrastructure described in Section V. We consider different network parameters such as latency and traffic conditions, i.e., TPS submitted to the system, in order to simulate typical operational characteristics.

A. Monitoring Indicators

To evaluate the system performance for each experimental test, we use the following indicators.

1) *Request round-trip time*, that is, the time elapsed from the moment in which a user submits an operation request (i.e., write) and the moment in which he/she receives service response. The monitoring procedure is accomplished by integrating a recording service both in the user and REST service.
2) *System resources utilization* in which we monitor the utilization of CPU and memory for all the related services. To keep track of the utilization of system resources, we relied on docker's built-in monitoring services.
3) *Transaction commit rate*, that is, the pace at which the transactions have been written in the blockchain. To do

TABLE II
DESCRIPTION OF THE EXPERIMENTAL TESTS CONSIDERING A BLOCKCHAIN NETWORK OF 28 NODES CONFIGURED WITH 4 MB/S NETWORK BANDWIDTH AND 3-MS LATENCY

Scenario No.	Description
s1	Clients generate 2.5 TPS towards the ordering service
s2	Clients generate 6 TPS towards the ordering service
s3	Clients generate 8 TPS towards the ordering service
s4	Clients generate 10 TPS towards the ordering service
s5	Clients generate 12 TPS towards the ordering service
s6	Clients generate 12.5 TPS towards the ordering service
s7	Clients generate 13.3 TPS towards the ordering service

so, we use the fetchblock[8] to monitor when transactions are recorded in the blockchain.

B. System Parameters and Configuration

Real environment characteristics (parameters and constraints) should be considered to assess system performance. In this work, we mainly study how 1) the number of TPS, 2) the operational (available) network bandwidth, and 3) the corresponding latency affect the system performance.

Based on the testbed architecture described in Section V, we have deployed 28 nodes, which represent the MSAs services in the context of SEED. The ordering service is configured on a "separate" system in order to provide the minimum failover mechanism required by Kafka, as described in Section V. Both the ordering service and MSA services were configured as virtual machines in the EPIC infrastructure, supporting in terms of hardware four CPU cores (1.7 GHz each) and 16 GB of RAM.

For all the test cases, a single *OR endorsing policy* was enforced, meaning that each transaction is endorsed only by one MSA node. Clients submit transactions to the MSA node in their jurisdictions (country) that would endorse those transactions. However, all the MSA nodes commit the transactions submitted by all the clients (MSA nodes are all in the same Hyperledger channel). Clients submit in total 130 k transactions, simulating the yearly volume of SEED, that are equally distributed to all the 28 MSA nodes of the network. In our experimental tests, client requests correspond to trader authorization records, as described in Section III, submitted in JSON format.

C. Throughput

In this set of experiments (see Table II), we study how the system performs under different traffic conditions when the bandwidth is limited to 4 Mb/s and the network latency is set to 3 ms. In particular, we scale the traffic targeting the ordering service in terms of TPS from 2.5 to 13 TPS.

Results demonstrate that end-to-end delay (request round-trip time) both for the end-user and the REST service is not impacted for the test cases up to 8 TPS. In fact, the average response time varies between 1 and 1.3 s. However, when the TPS rate increases

[8][Online]. Available: https://github.com/cendhu/fetch-block

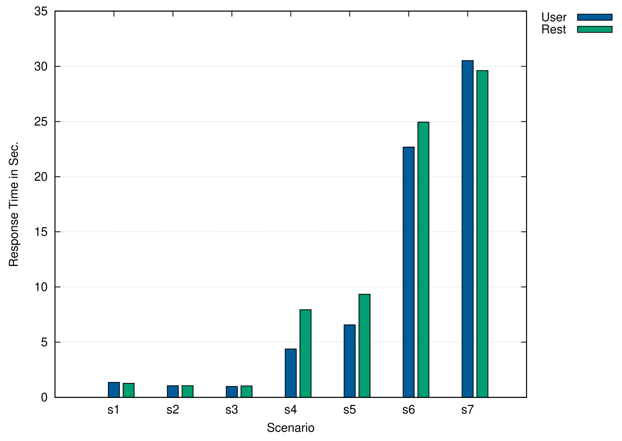

Fig. 5. User and REST service response time considering different number of TPS.

TABLE III
THROUGHPUT ACHIEVED WITH 4 MB/S BANDWIDTH AND RELATED RESOURCE UTILIZATION

Submission Rate per sec.	Average Commit Rate per sec.	Average Roundtrip time in sec.
2.5	2.474	1.345
6.25	6.106	0.952
8	7.774	0.884
10	9.642	0.453
12	11.443	7.264
12.5	11.921	25.533
13	5.826	28.566

to more than 10 TPS then the average response time increases up to 30 s, approximately 29× compared to the previous cases. This trend is confirmed by the REST service response time that follows the same trend. Fig. 5 overviews the round-trip time for serving a single request both for the end-user and the REST service as well.

Furthermore, to identify whether the request submission rate, i.e., TPS is influenced by the deployed configuration we also monitor the transactions commit rate. The commit rate increases with the TPS up to a certain threshold. After this threshold, the commit rate drops and the round-trip times grow considerably, indicating a poor performance in terms of transaction writing throughput. Table III lists the relationship between transactions submission, commit rate, and round-trip time.

Regarding resources utilization for MSA nodes, the CPU utilization for the peer service ranges between 33% and 41%. For instance, in the case of 2.5 TPS the CPU utilization approximates 35% of the available resources, which slightly varies for the rest of the cases. On the contrary, the REST service CPU utilization increases "steadily" as the TPS rate raises. In fact, CPU utilization from 2.5 to 13 TPS increases by 7×. This shows the stress of the REST service that corresponds to the delay introduced in the requests delivery. A similar trend is demonstrated by the CouchDB service as well, however, after 12.5 TPS the CouchDB CPU utilization starts to decrease because the REST service is

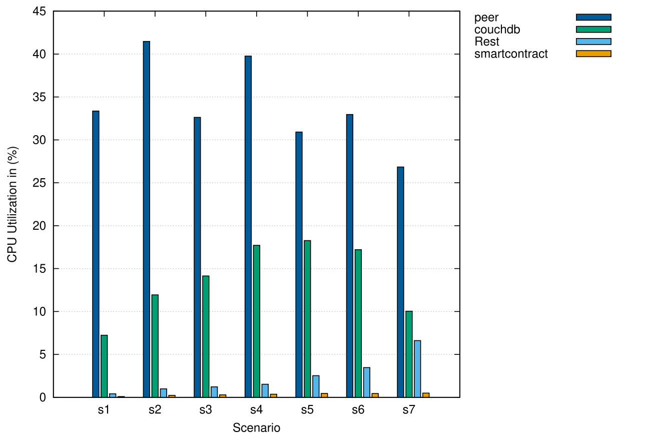

Fig. 6. MSA nodes services CPU utilization considering different number of TPS.

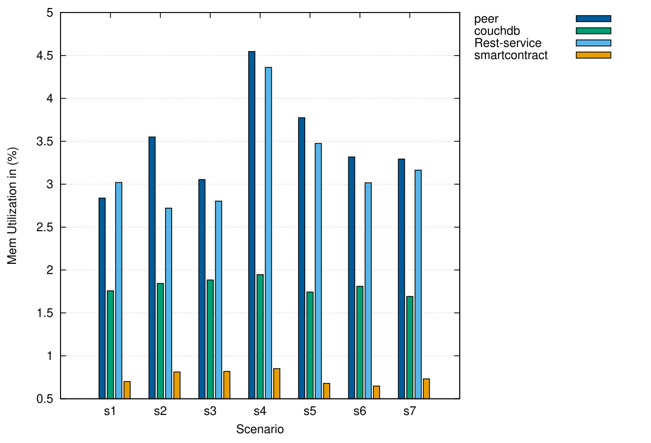

Fig. 7. MSA nodes services MEM utilization considering different number of TPS.

not capable of processing properly the TPS generated by the clients. As for the smart contract CPU utilization, it is negligible. Fig. 6 overviews the CPU utilization trend for MSA nodes hosted services.

As far as memory utilization for the peer and the REST service is concerned, it can be as low as 4.5%, whereas the memory footprint for the CouchDB remains stable under all cases (less than 2%). Even if the memory utilization by the smart contract increases, it is less than 1% in all tests. At this point, it should be noted that for all the MSA services, after 10 TPS the memory utilization decreases and this is due to the fact that the network bandwidth and the REST service throttle the generated traffic. Fig. 7 overviews the memory utilization trend for MSA services.

Regarding resource utilization by the ordering service, the number of TPS does not have any influence on the ZooKeeper service in terms of memory and CPU utilization as Figs. 8 and 9 demonstrate. However, this is not the case for the Orderer and Kafka services. Indeed, as TPS increases, the ordering service CPU utilization increases up to 2.5×. The same trend is followed also by the Kafka service, though after 10 TPS the CPU utilization decreases.

Overall, considering the current configuration for these scenarios, such an architecture cannot properly handle traffic more

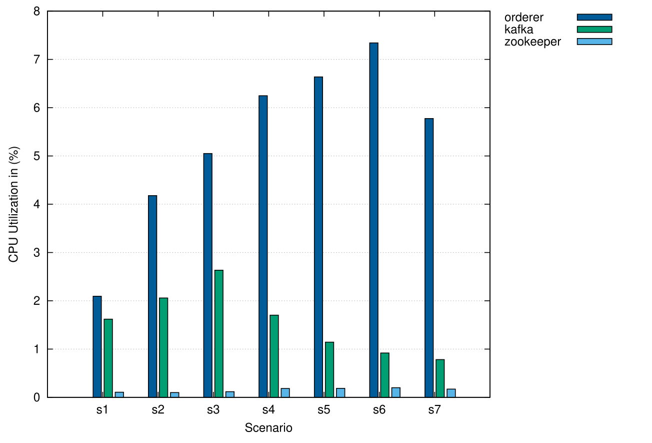

Fig. 8. Ordering services CPU utilization considering different number of TPS.

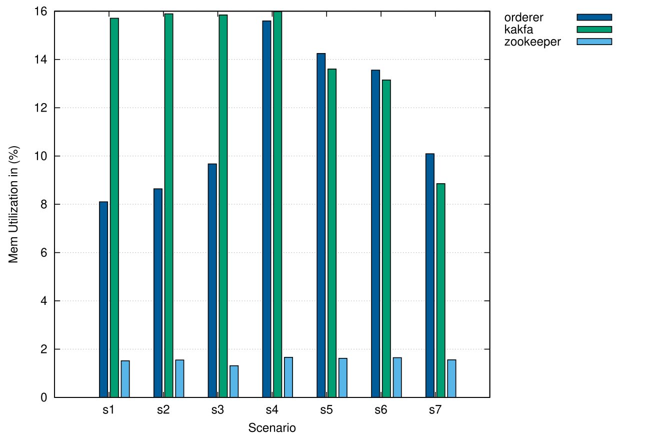

Fig. 9. Ordering services memory utilization with considering different number of TPS.

TABLE IV
DESCRIPTION OF THE EXPERIMENTAL TESTS CONSIDERING A BLOCKCHAIN NETWORK OF 28 NODES CONFIGURED WITH 4 MB/S NETWORK BANDWIDTH CONSIDERING DIFFERENT LATENCY VALUES

Scenario No.	Description
s1	The network latency is configured at 100 ms
s2	The network latency is configured at 300 ms
s3	The network latency is configured at 500 ms
s4	The network latency is configured at 1000 ms
s5	The network latency is configured to real distribution values

In all the scenarios, clients send 8 TPS to the ordering system.

than 10 TPS, as round-trip time for serving a single request increases "exponentially." The REST service is a point of throttling for the generated traffic, taking into consideration the limitation of the available bandwidth as well.

D. Network Latency

In this set of experiments (see Table IV), we analyze how network latency can affect service performance. In particular, the network bandwidth is limited to 4 Mb/s and the network latency for all the MSAs is configured to different values, i.e.,

TABLE V
NETWORK LATENCY DISTRIBUTION BETWEEN BLOCKCHAIN NODES

Delay in ms.	Number of Nodes
50	3
100	9
200	6
300	2
400	1
500	1
1000	2
2000	3
3000	1

TABLE VI
SYSTEM PERFORMANCE FOR DIFFERENT LATENCY VALUES

One-way latency (ms)	Submission Rate per sec.	Average Commit Rate per sec.	Average Roundtrip time in sec.
100	8	7.726	1.989
300	8	7.731	3.206
500	8	7.423	4.410
1000	8	7.470	6.658
(mixed)	8	7.743	2.035

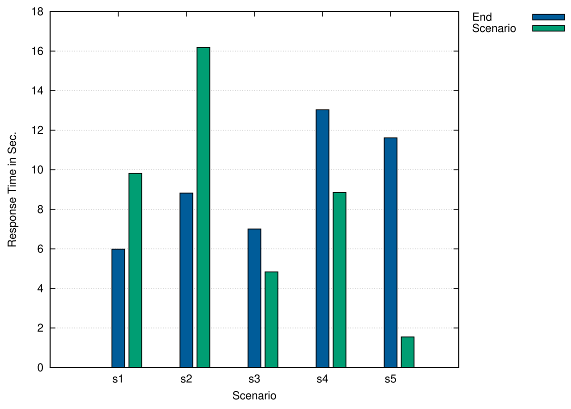

Fig. 10. User and REST service response time considering different network latency values.

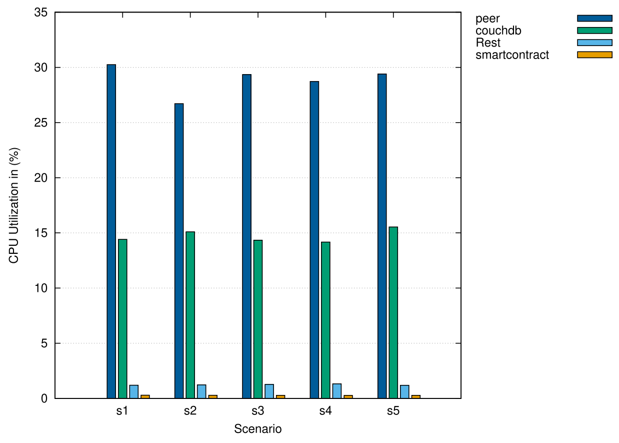

Fig. 11. MSA nodes services CPU utilization considering different network latency values.

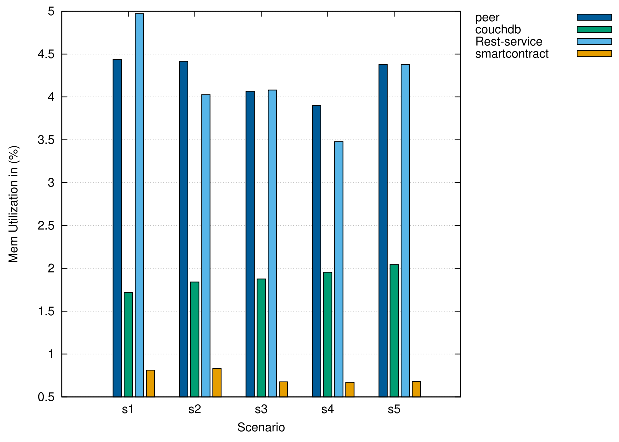

Fig. 12. MSA nodes services utilization considering different network latency values.

3, 100, 500, and 1000 ms. We also consider a test case in which MSAs nodes network latency is configured to real distribution values (see Table V) based on operational data. For all the tests, the traffic targeting the ordering service is configured at 8 TPS. This is an optimal value when the system is configured at 4 Mb/s, as the results in the Section VI-C demonstrate.

Results show that network latency highly affects end-user experience in terms of service responsiveness. For instance, when network latency is set to 3 ms, the request round-trip time is on average around 1 s, whereas as latency increases from 100 to 1000 ms the round-trip time increases up to 9×. This trend is illustrated in Fig. 10.

Furthermore, to identify whether the request submission rate, i.e., TPS is influenced by the latency we also monitor the transaction commit rate as in the case of throughput. In fact, the request submission rate and transactions commit rate keep almost the same pace under all latency test cases (see Table VI).

As far as resource utilization (both for CPU and memory) is concerned, neither the MSA nodes nor the ordering services are affected by network latency as Figs. 11–14 demonstrate correspondingly.

E. Bandwidth

In this set of experiments (see Table VII), we examine whether the provided bandwidth could be a constraint in the context of the provided service. We analyze how the system performs considering different network bandwidth, i.e., 1 Mb/s, 4 Mb/s, and 1 Gb/s, with an optimal network latency setting (that is 3 ms).

Results demonstrate that the available bandwidth speeds can highly influence system performance mainly in terms of resource utilization considering the maximum TPS among the available

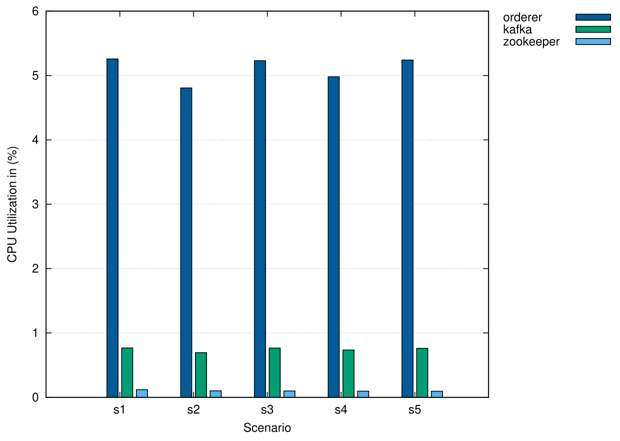

Fig. 13. Ordering services CPU utilization considering various latency values.

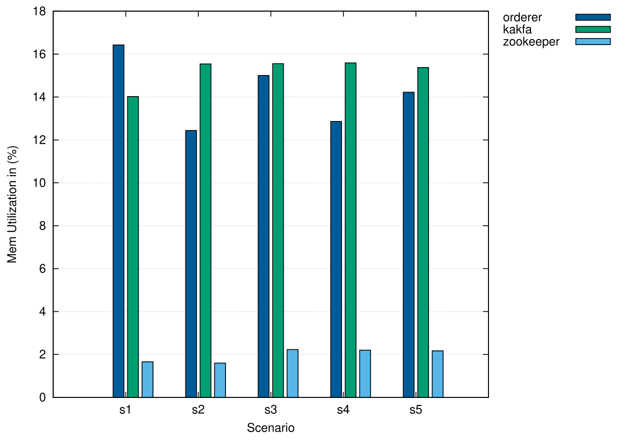

Fig. 14. Ordering services utilization considering various latency values.

TABLE VII
DESCRIPTION OF THE EXPERIMENTAL TESTS CONSIDERING A BLOCKCHAIN NETWORK OF 28 NODES CONFIGURED WITH NETWORK LATENCY OF 3 MS CONSIDERING DIFFERENT VALUES OF BANDWIDTH

Scenario No.	Description
s1	The network bandwidth is set to 1 Mbps
s2	The network bandwidth is set to 4 Mbps
s3	The network bandwidth is set to 1 Gbps
s4	The network bandwidth is set to 1 Gbps

bandwidth. In fact, the maximum supported TPS for 1 Mb/s, 4 Mb/s, and 1 Gb/s is 1, 12, and 47 TPS, respectively (see Table VIII and Fig. 15). This outcome shows that the available bandwidth influences the maximum number of TPS that can be sent to the ordering service. Interestingly, MSA nodes and ordering service resources utilization, with emphasis on CPU utilization, increases for all the services as TPS and available bandwidth grow (see Figs. 16 and 17). With reference to memory utilization, this trend occurs only for MSAs node peer and REST services (see Figs. 18 and 19).

TABLE VIII
MAXIMUM THROUGHPUT ACHIEVED WITH DIFFERENT BANDWIDTH VALUES

Bandwidth	Submission Rate per sec.	Average Commit Rate per sec.	Average roundtrip time in sec.
1 Mbit/s	1	0.996	1.634
4 Mbit/s	12	11.435	7.264
1 Gbit/s	48.5	47.870	1.362

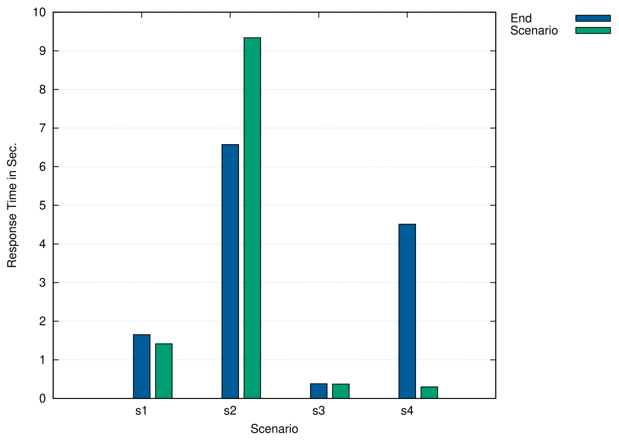

Fig. 15. User and REST service response time considering different bandwidth speeds.

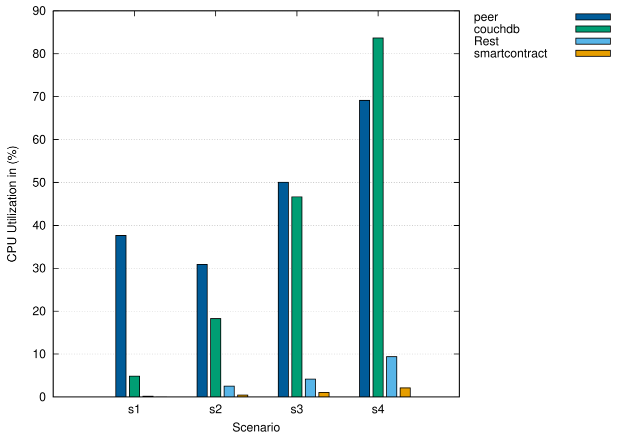

Fig. 16. MSA nodes services CPU utilization considering different bandwidth speeds.

F. Scalability

In the previous set of experiments, we analyzed how the system performs under various traffic conditions considering specific network configurations. However, at this point one might raise a concern about system scalability when the network size increases. To identify to what extent such a concern could be a burden in the implementation of SEED like b-government services, we run a set of experiment with different network sizes up to 28 nodes (see Tables IX, X, and II).

Figs. 20, 21, and 5 show end-users and REST services response time when the network size is configured with eight, 16, and 28 nodes. Results demonstrate that the number of nodes

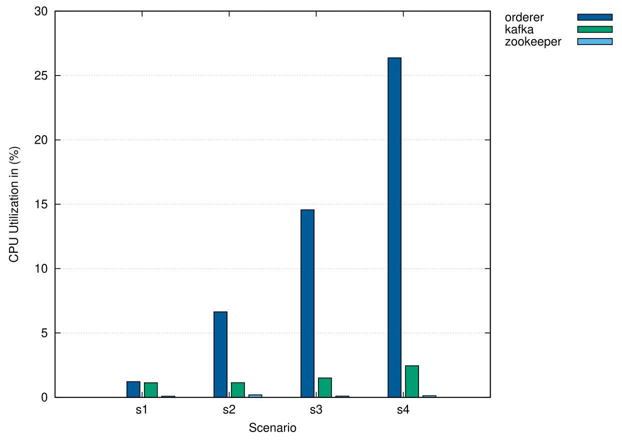

Fig. 17. Ordering services CPU utilization considering different bandwidth speeds.

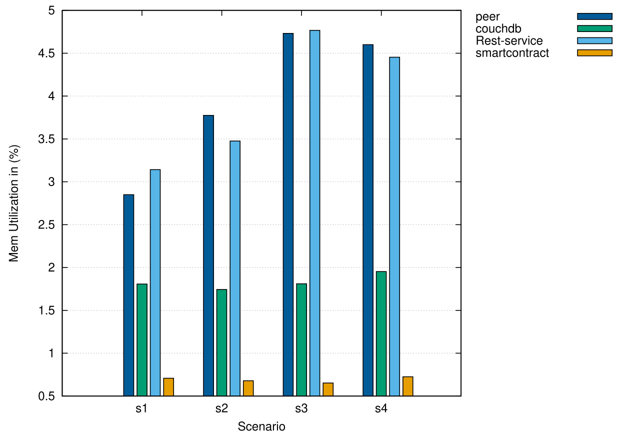

Fig. 18. MSA nodes services memory utilization considering different bandwidth speeds.

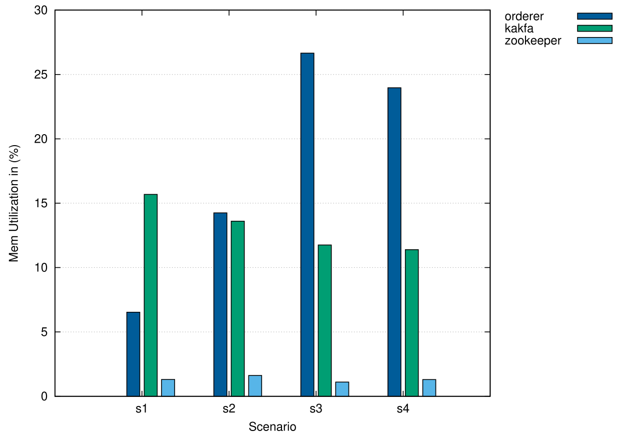

Fig. 19. Ordering services memory utilization considering different bandwidth speeds.

affects the network performance in terms of service quality, i.e., round-trip response time.

In particular, when the blockchain network is configured with eight nodes, the system resources start to saturate over 30 TPS; the user round-trip response time approximates 10 s for the user and 2 s for the REST service. In the case of a blockchain of

TABLE IX
DESCRIPTION OF THE EXPERIMENTAL TESTS CONSIDERING A BLOCKCHAIN NETWORK OF EIGHT NODES WITH 4 MB/S NETWORK BANDWIDTH AND 3-MS LATENCY

Scenario No.	Description
s1	Clients generate 10 TPS towards the ordering service
s2	Clients generate 12.5 TPS towards the ordering service
s3	Clients generate 15 TPS towards the ordering service
s4	Clients generate 20 TPS towards the ordering service
s5	Clients generate 30 TPS towards the ordering service
s6	Clients generate 35 TPS towards the ordering service
s7	Clients generate 40 TPS towards the ordering service

TABLE X
DESCRIPTION OF THE EXPERIMENTAL TESTS CONSIDERING A BLOCKCHAIN NETWORK OF 16 NODES WITH 4 MB/S NETWORK BANDWIDTH AND 3-MS LATENCY

Scenario No.	Description
s1	Clients generate 10 TPS towards the ordering service
s2	Clients generate 15 TPS towards the ordering service
s3	Clients generate 20 TPS towards the ordering service
s4	Clients generate 25 TPS towards the ordering service
s5	Clients generate 30 TPS towards the ordering service

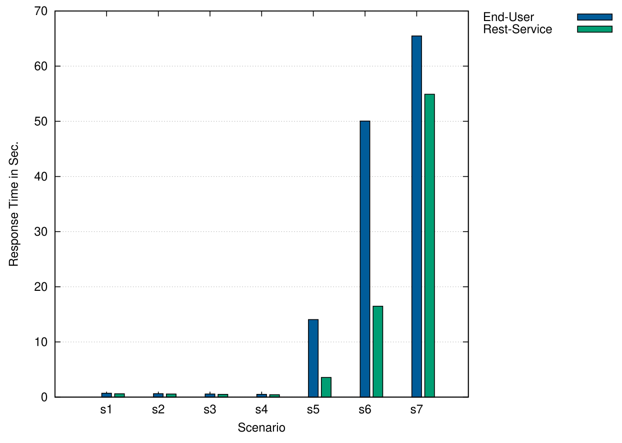

Fig. 20. User and REST service response time when the blockchain network is consisted of eight nodes.

16 nodes, the system resources saturate with less than 20 TPS, as the round-trip response time for both the user and the REST service is 30 s at 20 TPS (see Fig. 21). As shown in Fig. 5, for a blockchain network of 28 nodes and a throughput of 13.3 TPS, the round-trip response time is 30 s. Therefore, it is evident that the blockchain network size for Hyperledger Fabric based solutions influences the maximum throughput that the provided service can handle.

VII. RELATED WORK DISCUSSION

In this section, we present related work on two main areas: 1) use of blockchains in the context of e-government applications and services, and 2) performance evaluations of blockchain

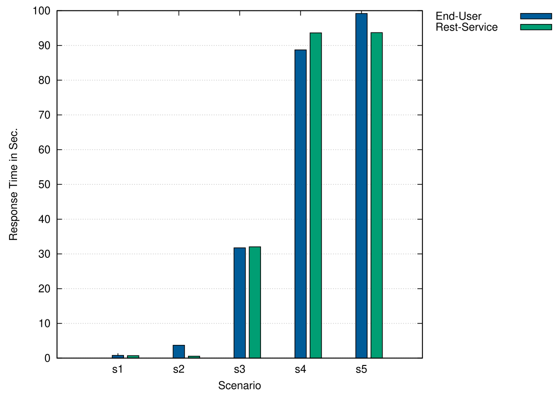

Fig. 21. User and REST service response time when the blockchain network is consisted of 16 nodes.

platforms. For both areas, we describe existing studies and compare them with our approach and performance evaluation results. To the best of our knowledge, we are the first to propose and evaluate a blockchain approach that supports collaboration across different entities in a network that consists up to 28 nodes.

Blockchain technologies have been identified as a promising technology to improve government services beyond their cryptocurrency use [26] in many countries [27] and in many areas including identity management, record keeping, value registry, voting systems, health records management, and agriculture [28]–[31].

A concrete example is described by Beris *et al.* in [32], where they propose a re-engineering of Diavgeia, the Greek government portal for open and transparent public administration using blockchain technologies. Their goal for using blockchain technologies is to make decisions taken by the government open, easily accessible to the public, and add immutability of all decisions over time. The trust model implemented by this work is different from the one adopted in this article since their focus is on decisions made by one government's institution, whereas in our architecture the focus is on interactions across MSAs in the EU.

Pongnumkul *et al.* [33] present a performance analysis and comparison of Hyperledger Fabric and a private deployment of Ethereum. According to their evaluation, the Hyperledger Fabric outperforms Ethereum, which is an expected result since Fabric relies also on an ordering service for consensus, whereas Ethereum uses proof of work. In their worst case scenario, when 10 000 transactions are submitted in parallel to both blockchain deployments, the latency of the Ethereum deployment is around 8 min, in contrast to 35 s for Hyperledger Fabric. Although the authors present a comprehensive evaluation including latency, transaction execution time, and throughput, they do not present the details about the mining difficulty set on their Ethereum deployment or the endorsement policy used in Hyperledger. A final remark is that both systems are not yet competitive in comparison to current database systems since their transaction rate is comparatively low. In contrast to our evaluation, although we focus only on the evaluation of Hyperledger Fabric, our results are more complete since we present additional details about the system architecture and the endorsement policies used. These parameters are of high importance since they can have a significant impact on the performance results.

Androulaki *et al.* in [19] describe in detail the Hyperledger Fabric platform, including the architecture components, applications, use cases, and a performance evaluation. They point out that Fabric is not yet performance-tuned, and that many parameters including transaction size, ordering service configuration, network architecture, among others may affect the results obtained. In their article, they focus on a simple evaluation simulating a cryptocurrency implementation with a single channel, an ordering service using Kafka (three ZooKeeper nodes and four Kafka brokers), and five peer nodes each belonging to different organizations. All nodes were colocated in the same data center with dedicated virtual machines (VMs). Using this configuration they performed a few experiments varying block size, CPU resources for each peer node, use of solid-state drive (SSD) versus RAM disks, increasing the number of peers in one data center, distributing peers over many different data centers. In our approach, we analyze the same parameters with a larger network of 28 nodes showing the behavior and performance issues when these nodes are also colocated but considering constraints in the network as well.

Thakkar *et al.* in [34] present a detailed performance analysis of Hyperledger Fabric, identify the main bottlenecks, and suggest optimizations. For the latter, the authors show empirical evaluation results comparing the performance before and after their optimizations are implemented. In summary, they show the following three main optimizations:

1) implementation of a caching mechanism for endorsement policy verification with an improvement;
2) parallel execution of endorsements;
3) optimization of the state database to a native approach that does not require REST invocations.

By combining their optimization approaches, in their specific scenario, the performance improvements were in the order of 16× for transaction throughput.

Gorenflo *et al.* in [35] also show how Hyperledger can be optimized for high transaction throughput. They suggest the following four main optimization approaches:

1) to remove from the ordering service the transaction data and only keep the metadata that is relevant for the transaction validation;
2) parallelization and caching of endorsement and validation peers;
3) caching of the state database using an in-memory approach;
4) splitting of endorser and committer roles to different peers.

Similarly to Thakkar *et al.* [34], these changes are architectural design choices that are supported by Hyperledger and do not require changes to the interfaces. By adopting these additional optimizations the authors achieve an increase in transaction throughput 7× higher than [34]. The optimizations proposed by Gorenflo *et al.* and Thakkar *et al.* show that the Hyperledger architecture is flexible and is prone to specific optimizations, considering a target blockchain architecture and use case, where high transaction throughputs can be achieved.

VIII. Conclusion

In this article, we described the results of an experimental activity aiming at studying the performance of an existing e-government service converted into its blockchain version. More in detail, we identified as suitable test use case an existing system (SEED) that is currently used within the EU to monitor the exchange of excise goods. The study included the conversion of the system into a *b-government* service considering different business requirements, and its deployment in our test infrastructure. We deployed SEED in our emulated infrastructure and tested under different network conditions in order to assess its performance and determine its limits.

This article showed that transforming an e-government data distribution service into a b-government equivalent is technically feasible considering specific hardware and network requirements. For instance, in a blockchain network consisted of 28 participants (MSAs) with a network bandwidth of 4 Mb/s a throughput of 8 TPS can be achieved without affecting service responsiveness, whereas in case of 1 Gb/s the throughput of the system rose to 48 TPS. Moreover, such a service transformation would benefit from blockchain built-in features such as data integrity and immutability, transparency, and common logic sharing. To the best of our knowledge, this is the first time that such a b-government service is evaluated under real network conditions.

In the near future, we plan to extend our work with the optimizations suggested by Thakkar *et al.* and Gorenflo *et al.*, and evaluate SEED performance over other blockchain platforms, i.e., Ethereum and study cybersecurity challenges on b-government services [34], [35]. Furthermore, in order to enhance trust, we will integrate a trusted execution environment, in the same way implemented by Mast *et al.* for database systems in [36], to the Hyperledger Fabric platform.

References

[1] F. Tian, "A supply chain traceability system for food safety based on HACCP, blockchain & Internet of Things," in *Proc. Int. Conf. Serv. Syst. Serv. Manage.*, Jun. 2017, pp. 1–6.

[2] D. Tse, B. Zhang, Y. Yang, C. Cheng, and H. Mu, "Blockchain application in food supply information security," in *Proc. IEEE Int. Conf. Ind. Eng. Eng. Manage.*, Dec. 2017, pp. 1357–1361.

[3] E. Tijan, S. Aksentijević, K. Ivanić, and M. Jardas, "Blockchain technology implementation in logistics," *Sustainability*, vol. 11, no. 4, p. 1185, Feb. 2019.

[4] V. D. Ndou, "E-Government for developing countries: Opportunities and challenges," *Electron. J. Inf. Syst. Developing Countries*, vol. 18, no. 1, pp. 1–24, 2004.

[5] R. Palanisamy and B. Mukerji, "Security and privacy issues in E-government," in *Cyber Behavior: Concepts, Methodologies, Tools, and Applications*. Hershey, PA, USA: IGI Global, 2014, pp. 880–892.

[6] A. Ramtohul and K. Soyjaudah, "Information security governance for e-services in Southern African developing countries e-government projects," *J. Sci. Technol. Policy Manage.*, vol. 7, no. 1, pp. 26–42, 2016.

[7] P. Papadopoulou, M. Nikolaidou, and D. Martakos, "What is trust in E-government? A proposed typology," in *Proc. 43rd Hawaii Int. Conf. Syst. Sci.*, Honolulu, HI, USA, 2010, pp. 1–10.

[8] M. Hammer and D. Shipman, "Reliability mechanisms for SDD-1: A system for distributed databases," *ACM Trans. Database Syst.*, vol. 5, no. 4, pp. 431–466, Dec. 1980.

[9] G. DeCandia *et al.*, "Dynamo: Amazon's highly available key-value store," *SIGOPS Oper. Syst. Rev.*, vol. 41, no. 6, pp. 205–220, Oct. 2007.

[10] A. Lakshman and P. Malik, "Cassandra: A decentralized structured storage system," *SIGOPS Oper. Syst. Rev.*, vol. 44, no. 2, pp. 35–40, Apr. 2010.

[11] M. Rauchs *et al.*, "Distributed ledger technology systems: A conceptual framework," *SSRN Electron. J.*, to be published, doi: 10.2139/ssrn.3230013.

[12] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, Jul. 1982.

[13] M. Jakobsson and A. Juels, "Proofs of work and bread pudding protocols (extended abstract)," in *Secure Information Networks*. Boston, MA, USA: Springer, 1999, pp. 258–272.

[14] S. King and S. M. Nadal, "PPCoin: Peer-to-peer crypto-currency with proof-of-stake," 2012. [Online]. Available: https://pdfs.semanticscholar.org/0db3/8d32069f3341d34c35085dc009a85ba13c13.pdf?_ga=2.189270443.1680383901.1589560842-407403572.1523881355

[15] D. Mingxiao, M. Xiaofeng, Z. Zhe, W. Xiangwei, and C. Qijun, "A review on consensus algorithm of blockchain," in *Proc. IEEE Int. Conf. Syst., Man, and Cybern.*, Oct. 2017, pp. 2567–2572.

[16] C. Cachin and M. Vukolic, "Blockchain consensus protocols in the wild," 2017. [Online]. Available: https://arxiv.org/abs/1707.01873

[17] European Commission, "Excise duties on alcohol, tobacco and energy," Sep. 2016.

[18] R. Oppliger, *SSL and TLs: Theory and Practice*, 2nd ed. Norwood, MA, USA: Artech House, 2016.

[19] E. Androulaki *et al.*, "Hyperledger Fabric: A distributed operating system for permissioned blockchains," in *Proc. 13th EuroSys Conf.*, 2018, pp. 30:1–30:15.

[20] C. Siaterlis, B. Genge, and M. Hohenadel, "EPIC: A testbed for scientifically rigorous cyber-physical security experimentation," *IEEE Trans. Emerg. Topics Comput.*, vol. 1, no. 2, pp. 319–330, Dec. 2013.

[21] J. Mirkovic, T. V. Benzel, T. Faber, R. Braden, J. T. Wroclawski, and S. Schwab, "The DETER project: Advancing the science of cyber security experimentation and test," in *Proc. IEEE Int. Conf. Technol. Homeland Secur.*, Nov. 2010, pp. 1–7.

[22] T. Benzel, "The science of cyber security experimentation: The DETER project," in *Proc. 27th Annu. Comput. Secur. Appl. Conf.*, 2011, pp. 137–148.

[23] C. M. Davis, J. E. Tate, H. Okhravi, C. Grier, T. J. Overbye, and D. Nicol, "SCADA cyber security testbed development," in *Proc. 38th North Amer. Power Symp.*, Sep. 2006, pp. 483–488.

[24] T. C. Eskridge, M. M. Carvalho, E. Stoner, T. Toggweiler, and A. Granados, "VINE: A cyber emulation environment for MTD experimentation," in *Proc. 2nd ACM Workshop Moving Target Defense*, 2015, pp. 43–47.

[25] K. E. Stewart, J. W. Humphries, and T. R. Andel, "Developing a virtualization platform for courses in networking, systems administration and cyber security education," in *Proc. Spring Simul. Multiconf.*, 2009, pp. 65:1–65:7.

[26] S. Ølnes and A. Jansen, "Blockchain technology as a support infrastructure in e-government," in *Electronic Government*, M. Janssen *et al.*, Eds. Cham, Switzerland: Springer, 2017, pp. 215–227.

[27] A. Ojo and S. Adebayo, "Blockchain as a next generation government information infrastructure: A review of initiatives in D5 countries," in *Government 3.0—Next Generation Government Technology Infrastructure and Services:Roadmaps, Enabling Technologies& Challenges*. Cham, Switzerland: Springer, 2017, pp. 283–298.

[28] A. Alketbi, Q. Nasir, and M. A. Talib, "Blockchain for government services—Use cases, security benefits and challenges," in *Proc. 15th Learn. Technol. Conf.*, Feb. 2018, pp. 112–119.

[29] N. Kshetri and J. Voas, "Blockchain-enabled e-voting," *IEEE Softw.*, vol. 35, no. 4, pp. 95–99, Jul. 2018.

[30] Y.-P. Lin *et al.*, "Blockchain: The evolutionary next step for ICT e-agriculture," *Environments*, vol. 4, no. 3, p. 50, Jul. 2017.

[31] E. Yavuz, A. K. Koç, U. C. Çabuk, and G. Dalkiliç, "Towards secure e-voting using Ethereum blockchain," in *Proc. 6th Int. Symp. Digit. Forensic Secur.*, Mar. 2018, pp. 1–7.

[32] T. Beris *et al.*, "Towards a decentralized, trusted, intelligent and linked public sector: A report from the Greek trenches," in *Companion Proc. World Wide Web Conf.*, 2019, pp. 840–849.

[33] S. Pongnumkul, C. Siripanpornchana, and S. Thajchayapong, "Performance analysis of private blockchain platforms in varying workloads," in *Proc. 26th Int. Conf. Comput. Commun. Netw.*, Jul. 2017, pp. 1–6.

[34] P. Thakkar, S. Nathan, and B. Vishwanathan, "Performance benchmarking and optimizing Hyperledger Fabric blockchain platform," in *Proc. IEEE 26th Int. Symp. Model., Anal. Simul. Comput. Telecommun. Syst.*, 2018, pp. 264–276.

[35] C. Gorenflo, S. Lee, L. Golab, and S. Keshav, "FastFabric: Scaling Hyperledger Fabric to 20,000 transactions per second," in *Proc. IEEE Int. Conf. Blockchain Cryptocurrency (ICBC)*, Seoul, South Korea, 2019, pp. 455–463.

[36] K. Mast, L. Chen, and E. G. Sirer, "Enabling strong database integrity using trusted execution environments," 2018. [Online]. Available: https://arxiv.org/abs/1801.01618