A Techno-Economic Study of Spectrum Sharing with Blockchain and Smart Contracts

Pedro J. Bustamante, Marcela M. Gomez, Martin B. H. Weiss, Ilia Murtazashvili, and Ali Palida

The authors analyze available system design options, implement a small-scale test scenario, estimate the implementation and usage costs, and demonstrate how these technologies impact spectrum sharing prospects.

Abstract

The wireless crunch resulted in excess demand for the use of spectrum, and spectrum sharing is increasingly being proposed as a solution. To date, little research has considered how blockchain technologies can enable greater spectrum sharing. To address this gap, we develop a stylized model to show how blockchains can be leveraged to facilitate the exchange of access rights on a well-known band. To demonstrate proof of concept, we analyze available system design options, implement a small-scale test scenario, estimate the implementation and usage costs, and demonstrate how these technologies impact spectrum sharing prospects. Our exercise shows that blockchains can alleviate some of the perceived obstacles to greater sharing of spectrum.

Introduction

Increasing demand for spectrum is responsible for the wireless crunch. Although greater exclusivity of rights is often prescribed to manage spectrum, technological developments and recent policy consider spectrum sharing as a way to increase spectrum access. To date, research on spectrum sharing has only scratched the surface on ways in which blockchain can improve prospects for spectrum sharing. We address this gap by considering whether blockchains provide cost-effective ways to improve spectrum sharing.

Specifically, we consider management of specific bands with the spectrum access system (SAS), a cloud-based service for managing communications of devices transmitting in the citizens broadband radio service (CBRS), expanded to a blockchain-based system. Our intention with this article is to make a first attempt at identifying transaction costs associated with blockchain adoption in an actual field setting, rather than to provide a general methodology for empirical blockchain analysis. We omit technical details regarding our empirical methodology to focus on conveying the qualitative insights we obtain from our analysis.

Spectrum sharing generally refers to *use it or share it* arrangements in which incumbent users (designated as primary users) are obligated to share their spectrum with new entrants (secondary users). For example, the U.S. Federal Communications Commission (FCC) facilitates spectrum sharing between federal incumbents and commercial users. This sharing agreement has been successful in many two-tier bands, including the advanced wireless service (AWS) band, which has inspired the development of a more flexible and multi-tier sharing approach for CBRS, the 3.5 GHz band [1]. The technologies in these examples use either explicit agreements that are developed in a centralized fashion (in the more static AWS case) or highly specialized (and expensive) coordination systems (in the more dynamic CBRS case).

Blockchain is a unique type of database in that its networks combine the following affordances: they are distributed (any party can contribute to the database and none control it), peer-to-peer, transparent, immutable, and self-executing in that algorithms and rules automatically trigger operations between nodes. Smart contracts, which are containers of code that can execute instructions on a blockchain when certain (usually external) conditions occur, enable additional applications of blockchain beyond cryptocurrency networks. The contracts are self-executing and can be hardwired with rules that constitute a system of governance for the blockchain network

Blockchains offer an enticing opportunity for spectrum management, including exclusive licensing (e.g.,establishing clear boundaries between allocations), unlicensed governance (e.g.,granular access control to spectrum bands), and the coordination of spectrum sharing frameworks [2]. Current sharing approaches have been criticized for their static and centralized structure (e.g.,the AWS band) and their complex and costly coordination structures (e.g.,the CBRS case). Blockchain-based applications could provide a cost-effective, decentralized, and dynamic alternative.

We build a simple smart contract-based spectrum sharing application for the 1695–1710 MHz band in the United States. (the AWS band), which has preestablished sharing arrangements and usage rights, as well as fixed incumbent sites. Our analysis complements an existing pilot program to explore the use of blockchain in spectrum management was undertaken by Agence Nationale des Frequences (ANFR). Since the operating costs and specific design considerations for this pilot project have not been published, our analysis offers complementary insight into blockchain deployments for spectrum sharing.

This article is organized as follows. The first section reviews the spectrum agreement governing the AWS band. Then we present an analysis of the available system design options. We continue

Digital Object Identifier:
10.1109/MCOM.001.2200317

Pedro J. Bustamante is with Carnegie Mellon University, USA; Marcela M. Gomez, Martin B. H. Weiss, Ilia Murtazashvili, and Ali Palida are with the University of Pittsburgh, USA.

by highlighting the main considerations of our proposed network. Finally, we present a small-scale implementation of the proposed system.

Spectrum Sharing in the AWS Band

Spectrum sharing in 1695–1710 MHz (the AWS band) is a two-tiered framework, where the incumbents are the meteorological satellites of the National Oceanic and Atmospheric Administration (NOAA). The incumbents utilize the band only for downlink transmissions (i.e., space-to-earth). The new entrants are LTE handsets or mobile stations (MSs) and are restricted to uplink operations (i.e., MS-to-base-station). The third set of participants are the base stations serving each MS, which act as coordination and connection points between PUs and SUs.

To protect the primary user (PU) from interference, regulators defined *restricted* zones. These areas, otherwise known as exclusion zones (EZs), are located around the PU and represent zones where the SUs are not allowed to transmit. A second area or coordination zone (CZ) was also defined as part of the rules to access the band. The CZ extends beyond the border of the EZ. Its boundary is defined according to several transmission factors (transmission power, antenna gains, propagation effects, etc.). Transmission privileges in the CZ are granted to secondary users (SUs) if, and only if, the proposed transmission will not contribute to the aggregate interference at the PU location in such a manner that it will become *harmful* [3].

Based on the concepts of the EZ and CZ, the authors have developed multiple approaches to specify the boundaries of these restricted areas. In this article, we utilize the notation introduced in [3] for the Multi-Tiered Incumbent Protection Zones (MIPZ), where three types of zones are defined:

- **No Access Zone (NAZ):** The spatial area surrounding the immediate vicinity of the PU, where transmission privileges are limited only to licensed incumbents
- **Limited Access Zone (LAZ):** The spatial area immediate to the NAZ, where a limited number of SUs are allowed to transmit simultaneously
- **Unlimited Access Zone (UAZ):** The region that lies outside the LAZ, where unlimited transmission privileges are granted to the SUs

As previously mentioned, the AWS band provides a suitable environment for testing blockchain-based coordination activities. First, the participants of the band are clearly defined and identified, which provides the basis to use different types of blockchain platforms. Second, the band represents the simplest example of spectrum sharing coordination, where one static PU interacts with well-known SUs via clearly defined communication channels.

Design Considerations for a Blockchain System

In this section, we discuss design alternatives of a blockchain-based system for spectrum sharing. The blockchain is characterized by diverse data types, platforms, governance systems, technical limitations, consensus algorithms, and so on. To design a blockchain-based spectrum sharing application, we focus on three main technical considerations: the type of data to be stored on the chain, the type of blockchain platform, and the corresponding consensus algorithm.

Type of Data

The data stored on the blockchain are usually referred to as *tokens* (i.e., digital, cryptographic, and exchangeable tokens). These can have several practical implications, such as representing access rights to some underlying economic value (e.g., property), permission to access some property, or the provision of services [2].

In most modern systems, two types of data are typically exchanged using the blockchain: native assets and external tokenized assets. Native assets usually refer to the *coin* being exchanged in the network (e.g., Bitcoin [BTC] or Ethereum [ETH]). On the other hand, *tokenization* of assets refers to the method utilized to *represent* real-world assets onto the blockchain. First, we need to translate property rights into assets with economic value. Then we can represent these assets as digital tokens (to be stored on a blockchain). Once assets are tokenized, users can start trading or exchanging them using the blockchain, while managing their economic value through smart contracts [2]. In the particular case of spectrum sharing applications, both native coins and tokens could be used.

Using Blockchain-Native Coins: In this scenario, we use a cryptocurrency platform and its native assets. The blockchain platform is used exclusively for payments in exchange for *spectrum units*. The PU receives some monetary crypto-units in exchange for access rights to the restricted zones around its transmitter. The *actual* exchange of access rights is managed through a separate system (e.g., an external spectrum exchange database). Another approach when using native assets is to map the value of the access rights to the native asset in the blockchain. Thus, we use a pricing function $f(x)$, which maps each spectrum unit to a price in terms of the native asset. For example, a spectrum access right would have some value in BTC or ETH.

Using Asset Tokenization: Instead of using the *coin* of the blockchain, the tokens representing *real-world* assets are directly stored on the chain. The first solution for spectrum sharing applications consists of a *notary* system on top of the blockchain. In this context, access rights are converted into property titles to be stored on the chain [4]. The agents can show *proof of existence* and *proof of ownership* of the different property titles representing transmission access rights.

The second approach is to directly virtualize or tokenize the electromagnetic spectrum resources (i.e., spectrum bands). There are multiple options to achieve the virtualization of spectrum. For example, the SUs in the band are 4G (LTE) users. This is very useful in terms of virtualization, given that LTE originated as a standard that enables more flexible spectrum allocation. The physical resource block (PRB) is the basic element for radio resource allocation in LTE. The number of allocated PRBs directly contributes to the bandwidth of a specific user. This utilization and aggregation of PRBs could be used as a rough proxy for a tokenized resource unit [5].

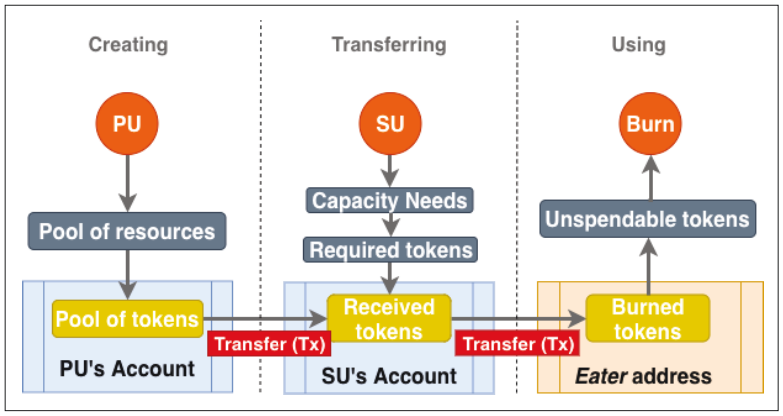

FIGURE 1. Transactions-only scenario.

Type of Blockchain Platform

Blockchain platforms are typically grouped by access rights into the general categories of public, consortium, and private blockchains [2] In public blockchains, any node (anonymously or through use of pseudonyms) can access the network. In spectrum applications, anyone could join the network and request a spectrum token (thus encouraging incentives for new participants to join the system). Public platforms are considered the only truly decentralized, democratized, and authority-free systems, which comes at the cost of such systems being less able to deter *bad* behaviors as a result of free entry.

In a private or permissioned platform, only a limited number of users, usually with known identities, have access to the system. A fully private blockchain would involve known identities of participants in the spectrum band, such as is the case with the 1695–1710 MHz band. In a private blockchain, an entity is in charge of network access control. Such an entity could be the PU or a sharing coordinator (the SAS in CBRS). Due to the nature of the nodes, the consensus algorithms, as well as the network assets, could be defined and modified by the network participants. Hence, any type of token could be stored in the chain while using *lightweight* consensus mechanisms. The downside of this option is the centralization of power.

In consortium platforms, instead of allowing any user to participate in the network or providing a single node with full access control, a few selected nodes are in charge of the most important functions of the system, including access control. In spectrum applications, multiple PUs and coordinators could be in charge of these activities. This approach also allows for native assets and a wider selection of consensus algorithms. In spectrum sharing, any type of blockchain platform could be utilized.

Type of Consensus Algorithm

To ensure reliability and correctness of the data stored on the blockchain, a *consensus algorithm* is used. Consensus algorithms have different computational, time, organizational, and energy characteristics where it is necessary to distinguish between the roles of full nodes and users. Both nodes and users can be any kind of device. Full nodes are required to store a full copy of the transaction history of the blockchain. Normally, full nodes also act as validators of transactions and creators of new blocks.

Cell phones are one example of the participants in the spectrum sharing scheme we consider. These devices are characterized by their limited resources (e.g., battery). In public blockchain algorithms using proof of work consensus algorithms, full nodes are required to perform resource-intensive tasks to create and validate new blocks. Consequently, if we consider a combination of any of these algorithms and all the participants acting as full nodes in our system, we would reach a sub-optimal solution [6]. Other algorithms such as proof of stake could also be problematic since the PU has the initial access rights over the majority of resources. This would result in unbalanced probabilities for validation of new blocks [6]. If we allow participants with limited resources to act as platform users rather than full nodes, their resource limitations could be neglected. In this scenario, the main limitation stems from other features of the platform, such as the delay introduced to validate blocks.

Many of the available private and consortium blockchains have opted to adapt lightweight consensus mechanisms, in particular, algorithms related to practical Byzantine fault tolerance (pBFT) [7]. The pBFT model focuses on creating a system that tolerates the presence of malicious nodes. Users are divided into two categories: clients and full nodes. These algorithms could also be utilized to coordinate a spectrum sharing system, where users with higher resource availability can act as full nodes, while other participants can behave as users sending requests to the full nodes.

Blockchain-Based Spectrum Sharing

In this section, we introduce the design of our small-scale implementation of blockchain-based spectrum sharing. We explore an application built on top of a public blockchain and a consortium-style solution to evaluate its performance and cost of implementation as a coordinator facilitator for spectrum sharing frameworks.

First, similar to the wireless tokenization approach presented in [5], we utilize PRBs as the underlying data. In our example, a token represents the right to utilize the assigned spectrum units (i.e., PRBs) for a given time. Although some tokens might indicate access to the same physical resources, their time feature converts them into unique, immutable, and unrepeatable assets.

Transactions

In a transaction-only scenario, we consider three steps: the creation of tokens by the primary user, how tokens are transferred, and how tokens are utilized by the SUs (Fig. 1).

Creating the Tokens: The PU is responsible for making the access rights available to the SUs. The tokens (and their rights) are made available via a resource pool. The process unfolds as follows: First, the PU determines the resources that are available at a given time. These resources are expressed as available PRBs for the LAZ and NAZ areas. Once the tokens are determined, they are stored in the blockchain. This allows the PU to transfer them to the SU.

Transferring the Tokens: The SUs need to acquire tokens to obtain transmission rights to

the band. The SU first translates its capacity needs into the number of required tokens. Afterward, an SU can request a PU to transfer these tokens through a common cryptocoin-like transaction.

Using the Tokens: Once an SU has exploited its transmission rights, the tokens must be *destroyed*. Hence, the SU must transfer the utilized tokens to an *eater* account, where they are no longer accessible. This idea comes from the concept of proof of burn, where tokens are *burned* by sending them to an unspendable address, known as an *eater* address [8].

Smart Contracts

Smart contracts provide an extra layer of management, supervision, and control of blockchain assets. In our design, we use two smart contracts that help us in the creation, transference, and usage of tokens.

Creation and Transfer of Tokens: The first smart contract helps with the creation of the pool of resources and its administration. The contract comprises four functionalities, as follows.

Register Pool of Resources: Allows the PU to register a list of the available tokens. It is important to note that smart contracts allow us to implement additional controls to improve the efficiency of the process. For instance, only the PU can call this function and hence register the available resources.

Register of SUs: Only registered SUs can receive the resources available in the pool. The goal of this function is to provide the PU and coordinators with a way to register verified SUs in the network. This is particularly useful to provide an extra layer of security on public platforms.

Check Availability: An SU can query the smart contract on the available tokens before requesting any resources.

Transfer the Resources: Allows the SU to obtain the tokens. SUs can request the smart contract to transfer a given number of tokens. If the request complies with the requirements, the smart contract transfers the tokens back to the SU.

Utilization of Tokens: Once the tokens have been registered, transferred, and utilized, they must be *destroyed*. The goal of the second smart contract is to include an additional layer in this *destruction* process through two main functions.

Use of Tokens: Automatically executed once an SU utilizes its assigned resources. The function transfers the used tokens from the SU's account into the smart contract's account.

Destruction* (Burn) *of Tokens: In the transactions-only scenario, once the tokens are utilized, they are immediately sent to the *eater* address. With the inclusion of a smart contract, these resources are temporally stored on the contract's account. Thus, multiple tokens can be sent for destruction at once. It is worth noting that only the PU can call or invoke this function.

Public Blockchain

The implementation of a blockchain-based spectrum sharing coordinator for our model system is depicted in Fig. 2. The base stations and PU act as full nodes. They have a full copy of the ledger, and they are allowed to participate in the main functions of the system. The MSs enter the network as general users due to their resource limitations. Their corresponding base stations act as their access points to the network.

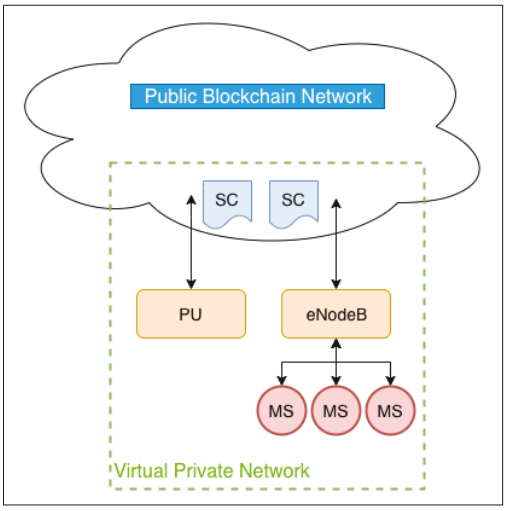

FIGURE 2. Public blockchain implementation.

To overcome the challenges associated with the *openness* of the public platform, some limitations are embedded in smart contracts. The users deploying the contracts are either the PU or the base stations. In this light, they become the *owners* of the contracts, which allows them to implement some security requirements (e.g., register the resources) mandated in the code of the smart contracts. Additionally, only registered SUs can invoke the smart contracts, thus creating a sort of *virtual, private subnetwork* within the public blockchain. Almost all public blockchains have only one native asset being exchanged in the network. Therefore, the system is required to implement a mapping function between the sharing tokens and the system's native asset.

Consortium Blockchain

A consortium system works as a *semi-private* mechanism that has a controlled user group, while also working across different organizations. Blockchain-based platforms are implemented as distributed ledgers; however, this ledger is usually composed of the blockchain itself and a *World State* database. The blockchain is a transaction log that stores all the records that have resulted in the current world state. The *World State* is a database that holds the current state of the ledger. These states are expressed as key-value pairs that can be created, updated, and deleted [9]. This structure improves the scalability and modular architecture in terms of the type of assets being exchanged and the available consensus algorithms.

The architecture of our consortium-like system is similar to the public blockchain implementation with some key differences (Fig. 3). First, the distributed ledger is only accessed by the members of the consortium. The agents with higher resources such as the PU act as entry points to the system for the SUs. The main functions of these *gatekeepers* include the management of users' identity, deployment of smart contracts, and access control to the channel. The inclusion of channels allows for different access control levels and operations in the system. For instance, users can create a channel to facilitate the exchange of assets

A consortium system works as a *semi-private* mechanism that has a controlled user group, while also working across different organizations. Blockchain-based platforms are implemented as distributed ledgers; however, this ledger is usually composed of the blockchain itself and a *World State* database.

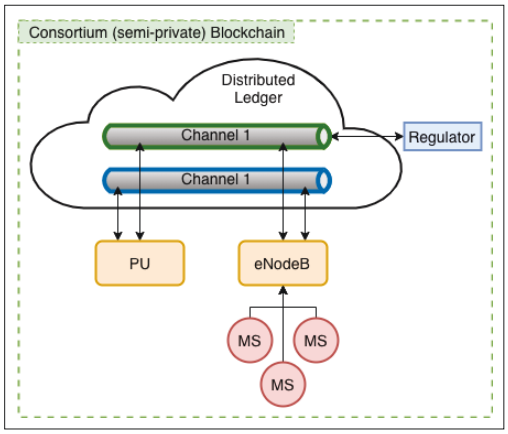

FIGURE 3. Semi-private blockchain implementation.

Contract's function	Transaction cost (gas)	Execution cost (gas)	Gas price (gwei).	Total cost (ether)	Price (USD)
Contract: createTransferSpectrumTokens					
deploy	358,571	—	20	0.00717142	17.21
registerSU	43,607	20,927	20	0.00129068	3.09
registerToken	42,224	20,760	20	0.00125968	3.02
transferToken	19,836	13,372	20	0.00066416	1.59
Contract: useBurnSpectrumTokens					
deploy	273,842	—	20	0.00547684	13.14
useToken	41,909	20,445	20	0.00124708	2.99
burnToken	39,879	33,607	20	0.00146972	3.52

TABLE 1. Local environment function usage.

Contract's operation	Transmission and execution cost (Gas)	Gas price (gwei)	Total cost (ether)	Price (USD)
Contract: createTransferSpectrumTokens				
deploy	453,390	1	0.000453390	1.08
registerSU	43,405	1	0.000043405	0.10
registerToken	42,745	1	0.000042745	0.10
transferToken	19,725	1	0.000019725	0.04
Contract: useBurnSpectrumTokens				
deploy	254,645	1	0.000254645	0.61
useToken	41,909	1	0.000041909	0.10
burnToken	39,901	1	0.000039901	0.09

TABLE 2. Testnet function usage.

and a different one to communicate transactions to the regulator.

Semi-private platforms provide a suite of consensus mechanisms. Many private platforms have opted to adapt and use pBFT-like algorithms [7]. In pBFT, users are divided into two categories: clients and full nodes. In our scheme, clients are the SUs in the system, while full nodes include the PU, the base stations, and/or a coordinator entity.

Small-Scale Implementation

In this section, we present a proof of concept of the proposed systems. We use blockchain and smart contracts to facilitate and coordinate spectrum sharing activities. Our goal is to illustrate the feasibility of our design and its corresponding cost and performance constraints. Since our model is for tutorial purposes, it also motivates additional consideration of the economic aspects of blockchain-based systems for managing the spectrum database.

Public Blockchain

One of the features of blockchains is that the technology is available for anyone to use, which has led them to be described as an example of knowledge commons: shared products of human knowledge that are available to all [10]. There are multiple public blockchain-based platforms available for the creation, development, and deployment of applications. Due to Ethereum's popularity, we test our system in both an Ethereum local environment and *Test-net*.

Since we rely on a working public blockchain, our system is based on the native asset and consensus algorithm of the platform. Thus, we include a mapping function in our smart contracts, which allows us to *translate* between *cryptocoins* and our *spectrum* tokens. The function we implement is a linear conversion that can be adjusted as a parameter in the contract. The default function converts 0.00001 units of Ether into the access rights to one PRB. A PU registers the available tokens as a variable in the contract (not as a native asset) and transfers some Ether to emulate these resources. Then, when an SU requests a set of tokens, it receives some of the Ether in the contract's account, thus emulating the transfer of tokens.

Local Public Environment: The first step in our experiment is to test the different functionalities in a local environment: a small implementation of Ethereum's nodes and accounts running locally on a workstation. We use *Ganache* and *Truffle* as development tools. The objective of this application is to verify the correct implementation of the functions that will be included in our smart contracts.

Deploying the Contract: We start by verifying that the contracts are correctly compiled. We validate that each of the functions and requirements is successfully transformed from contract scripts to Ethereum Virtual Machine (EVM) bytecode. The results of the costs of the deployment of our contracts are depicted in Table 1, where we show that both contracts are correctly initiated and deployed, and their associated gas cost (i.e., the Ethereum fee, denominated in ETH, to execute a transaction). In Tables 1 and 2, the gas price reflects the historical median price in Ethereum, while the total cost reflects the average price of Ether over the last 52 weeks.

Functionality Testing: Once the contracts are compiled and deployed on top of the local blockchain network, the next step is to test the different functions included in the system. For this purpose, automatic scripts are designed and deployed. Each script emulates a combination of invocations of the smart contracts' functions (e.g., register a pool of resources). Note that all the experiments we designed were successfully executed by the system. Besides verifying the correctness of our system functions, this set of tests allows us to calculate the costs associated with the execution of the different functions, as shown in Table 1.

Ethereum Testnet: Next, we deploy our proof of concept on a working Ethereum testnet. We

choose the *Ropsten Test Network*, *Remix*, and *Metamask* as development tools. Our testing process is divided into two phases.

Deploying the Contract: Both smart contracts are successfully transferred to and deployed in the *Ropsten* network, where they are publicly available to any user in the network. However, as previously mentioned, only registered users can invoke the contract functionalities.

Functionality Testing: After the contracts are deployed, we can test their functionalities. Table 2 summarizes the results obtained in the tests we ran. Similar to our local development, the deployment and usage of smart contracts imply a cost of operation. These costs reflect the actual fees charged by the *Ropsten* network nodes.

Another important measure in the creation of our system is the latency introduced by the nature of the public platform and its consensus mechanisms. In our experiments, the average time to add the block containing our smart contracts' transactions is around 15.3 s. Indeed, most of the blocks were confirmed in less than 20 s (Fig. 4) for all our experiments in the Ethereum Testnet.

Conclusions

Research on spectrum increasingly considers sharing as well as opportunities for greater decentralization of management of spectrum [11]. We implemented a small-scale model on two well-known blockchain-based platforms to better understand the implications of developing such applications for spectrum management. Most significantly, our exercise illustrates that conventional SAS systems, which are centralized, are not the only way to implement sharing arrangements.

As with any tutorial-style proof of concept, our model has limitations, such as the assumption that participants in the band are well known and that the spectrum band can be virtualized with a native token. In addition, real-time solutions are constrained by the validation and confirmation times of the underlying platform. Further analysis of these issues is necessary. What our research suggests is a fruitful path for further exploration, which we think worthwhile given increasing attention to sharing as a way to alleviate the wireless crunch.

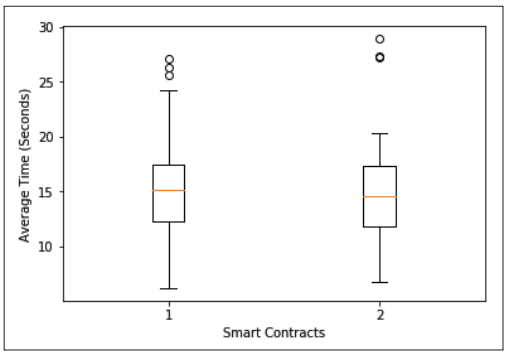

FIGURE 4. Average confirmation time.

References

[1] F. Hu, B. Chen, and K. Zhu, "Full Spectrum Sharing in Cognitive Radio Networks Toward 5G: A Survey," *IEEE Access*, vol. 6, 2018, pp. 15,754–76.

[2] M. B. Weiss *et al.*, "On the Application of Blockchains to Spectrum Management," *IEEE Trans. Cognitive Commun. and Networking*, 2019.

[3] S. Bhattarai *et al.*, "Defining Incumbent Protection Zones on the Fly: Dynamic Boundaries for Spectrum Sharing," *2015 IEEE Int'l. Symp. Dynamic Spectrum Access Networks*, 2015, pp. 251–62.

[4] C. Sullivan and E. Burger, "E-Residency and Blockchain," *Computer Law & Security Review*, vol. 33, no. 4, 2017, pp. 470–81.

[5] M. Gomez, L. Cui, and M. B. Weiss, "Trading Wireless Capacity Through Spectrum Virtualization Using LTE-A," 2014 TPRC Conf. Paper, 2014.

[6] L. S. Sankar, M. Sindhu, and M. Sethumadhavan, "Survey of Consensus Protocols on Blockchain Applications," *2017 4th Int'l. Conf. Advanced Computing and Commun. Systems*, 2017.

[7] L. Bach, B. Mihaljevic, and M. Zagar, "Comparative Analysis of Blockchain Consensus Algorithms," *2018 41st Int'l. Convention on Info. and Commun. Technology, Electronics and Microelectronics*, 2018, pp. 1545–50.

[8] X. Xu *et al.*, "The Blockchain as a Software Connector," *2016 13th Working IEEE/IFIP Conf. Software Architecture*, 2016, pp. 182–91.

[9] E. Androulaki *et al.*, "Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains," *Proc. 13th ACM EuroSys Conf.*, 2018, p. 30.

[10] I. Murtazashvili *et al.*, "Blockchain Networks as Knowledge Commons," *Int'l. J. Commons*, 2022.

[11] P. Bustamante *et al.*, "Spectrum Anarchy: Why Self-Governance of the Radio Spectrum Works Better Than We Think," *J. Institutional Economics*, vol. 16, no. 6, 2020, pp. 863–82.

Biographies

Pedro J. Bustamante (pbustamante@cmu.edu) received his Ph.D. in information sciences from the University of Pittsburgh. He is currently an assistant teaching professor at Carnegie Mellon University. His research interests mainly lie in the areas of telecommunications management, spectrum sharing, blockchain, and governance.

Marcela M. Gomez (mmg62@pitt.edu) received her Ph.D. in information sciences from the University of Pittsburgh. She is currently the senior data analyst for the office of the Senior Vice Chancellor for Research at the University of Pittsburgh. Her research interests include data analysis, secondary spectrum markets, virtualization, and governance.

Martin B. H. Weiss (mbw@pitt.edu) is a professor in the Department of Informatics and Networked Systems in the School of Computing and Information and associate director of the Center for Governance and Markets at the University of Pittsburgh as well as a research partner at SpectrumX, an NSF Spectrum Innovation Institute. His current research focus is on dynamic spectrum access and intelligent wireless systems.

Ilia Murtazashvili (ilia.murtazashvili@pitt.edu) is an associate professor in the Graduate School of Public and International Affairs and associate director of the Center for Governance and Markets at the University of Pittsburgh as well as a research partner at SpectrumX. His research focuses on governance of new commons (technology) and old commons (natural resources).

Ali Palida (afp31@pitt.edu) received his Ph.D. in economics from the Massachusetts Institute of Technology in 2020 and is currently a postdoctoral research with the Center for Governance and Markets at the University of Pittsburgh. His research interests lie in the economics of organizations as well as more general questions related to the governance of value creation in organized settings.