IoTShare: A Blockchain-Enabled IoT Resource Sharing On-Demand Protocol for Smart City Situation-Awareness Applications

Bechir Hamdaoui, Mohamed Alkalbani, Ammar Rayes[†], and Nizar Zorba[‡]
Oregon State University, Corvallis, OR 97331, hamdaoui,alkalbmo@oregonstate.edu
[†] Cisco Systems, San Jose, CA 95134, rayes@cisco.com [‡] Qatar University, Doha, Qatar, nizarz@qu.edu.qa

***Abstract*—We propose a Blockchains-based, distributed protocol for enabling the deployment of IoT networks on-demand on top of IoT devices. Specif cally, the proposed protocol leverages Blockchains technology to: (i) enable distributed and secure authentication, registration, and management of participatory IoT devices; (ii) provide fast discovery of IoT resources and scalable and secure instantiation of IoT networks-on-demand; and (iii) manage payment operations and ensure reliable fund transfers among the network entities. The proposed protocol relies on a peer-to-peer network communication infrastructure to allow communications among the IoT devices in a distributed manner, and uses a self-recovery/self-healing mechanism to ensure robustness against device failure and maliciousness. The protocol also introduces and uses a reputation system to monitor registered devices to keep track of their service delivery quality, so that their service delivery reputations could be leveraged for future device selection and mapping. We implemented and evaluated the proposed protocol intensively using simulations, and showed that it scales well with network parameters, is resilient to faulty devices, and is robust to 51% attack.**

***Index Terms*—Smart Cities, IoT Networks, Blockchains.**

I. Introduction

The proliferation of Internet of Things (IoT) devices emerges as a key enabler for a new era of city services and applications [1]. These city services and applications, ranging from realtime surveillance and precision health to city traff c control and emergency management, can be enabled via carefully-chosen collections of geographically distributed IoT devices, with sustained Internet connectivity, that interactively execute specif c tasks as dictated by the underlying applications they support [2]. Such collections of IoT devices, forming what we term here *participatory IoT networks-on-demand* (or `NoD`) instances, are created dynamically, on-demand upon end users' requests.

Albeit its great potential in enhancing city service responsiveness, the enabling of this participatory IoT networks-on-demand paradigm requires new, innovative networking solutions that can overcome key challenges arising from the increased network connectivity demands due to the large numbers of IoT devices and from the devices' limited power and computing resource capability. Fortunately, new networking technologies are being emerged as potential solutions to these challenges. For instance, cognitive radio (e.g., [3, 4]) and mmWave (e.g., [5, 6]) network access are being adopted to tackle the wireless connectivity and bandwidth challenges, whereas other technologies such as in-network caching (e.g. [7–9]) and edge cloud off oading (e.g. [10, 11]) are emerging as potential solutions for overcoming the resource limitation problems of these IoT devices.

This work was supported in part by Cisco Systems and the US National Science Foundation (NSF) under NSF award No. 1162296 and 1923884.

In addition, blockchains technology [12], the main technology behind Bitcoin, emerged as a potential method for decentralizing the recordkeeping of digital currency transactions. It allows Bitcoin users to transfer funds, validate transactions and record information in a fully distributed manner without the need for any intermediary party. All transactions are recorded into blocks, verif ed by all users, and added to the blockchains. Users use public keys as their identities to provide anonymity and privacy, and any user can choose to be responsible for mining and adding blocks to the blockchain ledger. Though has initially been used for cryptocurrency, due to its distributed nature and great ability in simplifying recordkeeping, blockchains has been adopted in many recent works to support IoT. In this paper, we propose a distributed resource allocation protocol that leverages blockchains technology to enable and ease IoT device resource sharing on-demand to support smart city applications and services.

A. Related Work: Blockchains for IoT Support

There have been recent works that focused on adopting blockchains technology for IoT support, and in this section, we go over some of the notable ones. Most of prior work focuses on securing IoT communication through the use of consensus among IoT devices, and on the use of blockchains as a method for storing data or system conf gurations. In [13], the authors discuss IoT and its presence in today's technology advancement and the need for more secure database management and data access. The authors dive into the limitations of IoT devices in terms of security, give a brief description of blockchain technology and its advantages, and propose a new method of altering existing blockchain technology to help cater IoT devices by proposing to include a shared ledger and move processing from IoT devices to central entities. The authors conclude that IoT technology is not fully ready to use blockchains. Unlike this work, our framework integrates blockchains with IoT to increase network security and scalability. In [14], the authors propose to integrate IoT, blockchains and cloud technologies altogether. Their methodology relies on pushing most of blockchains processing to the cloud, while still allowing IoT devices to connect to the cloud but for authentication purposes. They show that latency presents the biggest challenge when it comes to integrating IoT and cloud services, and propose to rely on local clusters as a better alternative. Our proposed framework touches on similar aspects in the sense that it also uses local clusters instead of centralized entities in the cloud to do the processing. However, our proposed protocol relies on edge devices to do the processing while limiting the processing threshold through the use of proof-of-stake (POS) instead of proof-of-work (POW) approach [15].

In [16], the authors discuss the advantages of integrating blockchains in the industrial sector to provide overall better business opportunities, visibility and transparency. The authors' proposal is to use blockchains as a method of keeping track of IoT devices data. This data is then federated to different agencies based on their keys in the blockchain. This allows for access to specif c entries in the blockchain all the time and to specif c agencies, thereby providing more transparency to the data collected while still keeping its privacy from other agencies. The authors in [17] propose the use of blockchains to keep track of IoT devices and their conf gurations and do so via Ethereum, which allows to write custom code on top of the blockchain and to use POS instead of POW. The authors also propose limiting the number of IoT devices needed in the POS round to minimize the communication overhead associated with the miner selection scheme. The authors in [18] propose a new protocol that leverages micro-controllers technology to enable the visualization of IoT devices. The authors argue that cloud centric virtualization mechanisms for IoT devices tend to experience high latency while moving the work to the edge devices yields better performance. The authors provide simulation to their approach in measuring the latency between two arduino devices connected to a wif network. The authors argue that using their approach lowers the overall latency, compared to using a cloud centric approach. They also discuss the possibility of integrating blockchains as part of the software on the edge devices, to enable more secure and scalable implementation of their protocol.

Our proposed framework differs from those existing works in that in addition to enabling distributed censuses among IoT devices, it provides a distributed method for reserving IoT resources on-demand, while at the same time adopting blockchains to increase scalability, security and robustness.

B. Our Contributions: `IoTShare`

We propose `IoTShare`, a distributed, blockchain-based protocol that enables the deployment of networks-on-demand (`NoD`) instances on top of participatory IoT devices. `IoTShare` leverages blockchains to enable distributed IoT resource sharing on-demand. Specif cally, it enables:

- Management and recordkeeping of participatory IoT devices. It does so by providing a mechanism that allows IoT devices to authenticate, join and register themselves to the network, and to broadcast and share their resource ID and characteristics (e.g., type, amount, location, duration, etc.) with the already registered devices.
- Distributed mappings of `NoD` requests on top of the registered IoT devices. It does so by providing a mechanism that allows the discovery of IoT devices that satisfy the requirements of `NoD` requests, and the mapping of the accepted requests on top of the discovered IoT resources.
- Service delivery verif cation by providing mechanisms that allow the monitoring of committed devices to ensure that they are meeting their delivery agreements. The mechanisms also allow for the building and maintaining of trust and reputation scores for devices, based on their delivered service quality to help f lter out malicious devices that do not perform their agreed upon tasks.
- Fast recovery from failed service delivery due to malicious (or non-malicious) device behaviors, where IoT devices intentionally or unintentionally fail to deliver agreed upon services. This is achieved by providing a self-recovery mechanism that allows to f nd quick replacements to failed devices upon their detection by the monitoring mechanism.
- Service rewarding through a mechanism that handles the payment to devices upon completing their service delivery. It essentially allows to check for fund availability and to transfer funds between different devices once service is successfully delivered.

The proposed protocol, `IoTShare`, also incorporates a reputation system to monitor and keep track of services delivered by registered IoT devices, to ensure that service delivery agreements are met. Using simulations, `IoTShare` is evaluated intensively to assess its scalability, its resiliency to faulty nodes, and its robustness to malicious behaviors.

The rest of the paper is organized as follows. In Section II, we describe the network infrastructure used by the proposed protocol. In Section III, we present `IoTShare`, the proposed IoT resource sharing on-demand protocol. In Section IV, we study and assess the effectiveness of the proposed protocol. In Section V, we discuss the security aspects and challenges pertaining to `IoTShare`. In Section VI, we highlight and discuss open research directions and challenges. Finally, we conclude the paper in Section VII.

II. System Architecture and Design Goals

We begin by describing the system model and architecture of the studied smart city to enable the deployment of participatory IoT networks on-demand. Then, we present the design goals and requirements of the protocol, `IoTShare`, introduced to enable distributed embedding of these IoT networks on-demand on top of participatory IoT devices. We f nally provide a brief background on blockchains technology for completeness since `IoTShare` is based on such an emerging technology.

A. System Model and Architecture

We consider a city-wide network constituted of many IoT devices spread all over the city, and a set of access points (also referred to as edge clouds) also spread across the city to provide Internet connectivity to the devices. An IoT device interested in joining the network needs to advertise, upon joining the network, its device characteristics or directory containing key information such as its resource type, resource availability, location, and bounty. The system allows devices to submit networks-on-demand (or `NoD`) requests, to be mapped on participatory IoT devices. A device submitting a request is referred to as *a consumer*. For example, an `NoD` request could be initiated to serve for tracking a target (e.g., malicious person) that is moving through different regions in the city. Each submitted `NoD` request is propagated to other IoT devices in the network using a peer-to-peer communication infrastructure to enable its mapping to the available IoT devices, as will be explained later. Our proposed protocol, `IoTShare`, uses blockchains for device management and request mapping onto IoT devices. For this, a set $\mathcal{M}$ of IoT devices are designated to serve as miners, which are responsible for handling fund transfer and device payment through creation and addition of blocks; more on this will be provided later. Each `NoD` request is submitted in the form of 5-tuple G=(V, $\mathcal{E}$, D, $\mathcal{C}$, B), with V specifying the number of requested nodes (also referred to as request size), $\mathcal{E}$ specifying the set of connections/edges between the requested nodes, D specifying the request duration, $\mathcal{C}= \{(Loc_1, Type_1, Cap_1), \ldots, (Loc_V, Type_V, Cap_V)\}$ specifying the location, the resource type and the resource capacity of each requested node, and B specifying the bounty associated with the request. Note that all these request parameters are to be determined by the application being supported by the `NoD` request, and are provided by the consumer initiating the request as input to `IoTShare`. We consider that the IoT devices are located randomly across the city, and that the city is divided into L different regions. When an IoT device joins the network, its characteristics will be added as an entry to `Directory`, a data structure that is updated every time a new device joins the network, and is stored and maintained by all devices through the blockchains, to be described later. Each entry of `Directory` contains the following device characteristics.

- *Device ID:* Serial number and/or IP/MAC addresses.
- *Resource Type(s):* sensing (video, temperature, traff c density), computation, and/or communication.
- *Location:* GPS location and region ID.
- *Availability:* Time period(s) during which device is available for participating in networks on-demand instances.
- *Capacity:* Amount/capacity of the resource available for sharing; i.e., computation (CPU power), sensing (sensed data per second), and/or bandwidth (bps).
- *Network access type:* Technology type used for connecting to the network; e.g., WiFi and/or cellular.
- *Bounty:* Cost of service.

B. Design Goals and Requirements

To enable networks-on-demand mapping on top of participatory IoT device resources, `IoTShare` must meet the following design goals and requirements:

a) `NoD` *request mapping:* `IoTShare` should provide a mechanism for each IoT device, willing to participate in networks-on-demand, to join the network, and to obtain all information needed to allow it to interact and communicate with other existing devices. It should also allow the mapping of `NoD` requests on top of participatory IoT devices.

b) Service delivery and integrity: `IoTShare` should provide monitoring mechanisms that ensure that the committed devices are performing their tasks and operations as agreed upon. The protocol must also provide a backup plan for recovering from service interruptions and service agreement violations resulting from device misbehavior and/or failure.

c) Service rewarding, payment and incentiveness: `IoTShare` should provide a mechanism that ensures that the served devices be rewarded for their offered services. With blockchains, this boils down to providing a secure mechanism for checking fund availability and for transferring funds between different devices. Making sure that devices are rewarded for their offered service will serve as great incentive for devices to join the network, participate on networks on-demand, and do and complete their tasks as agreed upon.

d) Scalability and distributivity: Due to the complexity, in size and numbers, of the system at hand, it is important that the different components of the protocol be scalable and implementable in a distributed manner whenever possible.

e) Security and fault tolerance: `IoTShare` should contain mechanisms that ensure the security of IoT devices, as well as the robustness against malicious behavior (intentional) and/or failure (unintentional) of devices during the performance of their assigned tasks.

C. Blockchains

`IoTShare` relies on blockchains technology [12] to allow scalable, distributed and fast management and mapping of the `NoD` requests. In `IoTShare`, a blockchains, BC, is used to act as a Turing State Machine to ensure data integrity and validity throughout the system, and contains a time stamped history of all transactions occurred in the system. For each mapped `NoD` request, transactions are added to the blockchains to contain and record all information about the accepted requests (e.g., the devices that choose to be part of the networks on-demand and serve as providers). Each device is associated with a wallet, W, where a wallet is a key pair of private and public keys. When a consumer issues a bounty B (the reward that each provider will receive as a payment for its offered service), a private key is used to sign that transaction. Later on, anyone in the network could use the signature as well as the public key to validate a said transaction. Each transaction is signed with a different signature to remove threat of double spending [19]. Miner nodes are in charge of collecting transaction entries and adding them to the blockchains, BC.

`IoTShare` does not adopt Bitcoin's proof-of-work (or POW) as the method for miner selection, due to its ineff ciency and the challenges it presents when used in the context of IoT. Such challenges arise from the limited storage and computation resources that IoT devices have, the delay sensitivity that applications to be supported by mapped IoT networks may have, and the limited bandwidths of the links (often wireless) that connect the IoT devices to the network. For these reasons, `IoTShare` uses the proof-of-stake (PoS) as its method for selecting miners [15]. This will be discussed later in the paper.

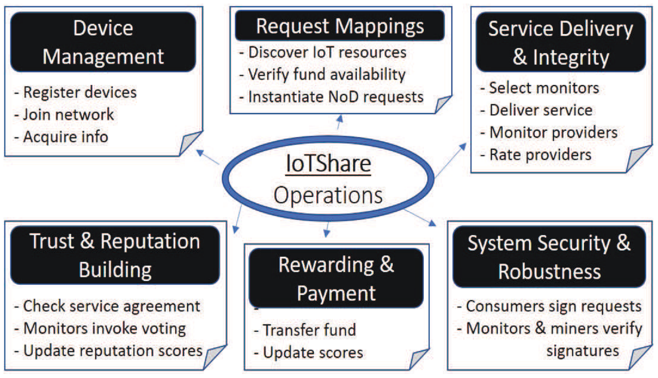


Fig. 1. `IoTShare` architecture with main operations

III. The Proposed Protocol: `IoTShare`

In this section, we present our distributed IoT resource sharing on-demand protocol, `IoTShare`, which is developed with the aim to meet the aforementioned design goals and requirements. Note that the terms device and node will be used interchangeably throughout to refer to the same thing.

A. `IoTShare` *functions*

At its highest level, `IoTShare` has the following operations, which are shown in Fig. 1.

a) Device management: To allow IoT devices to participate in networks on-demand, `IoTShare` provides a mechanism for these participatory devices to register with the network, and acquire information about other registered devices so they can join and form peer-to-peer networks with existing devices. Upon joining the network, a node connects to some other nodes already joined the network and requests a copy of `Directory` and the blockchains, BC. It also broadcasts its device information to the other nodes in the network.

b) `NoD` *request mapping:* This phase of the protocol provides a mechanism to allow the discovery of IoT devices satisfying the requirements of the requests, such as availability, capacity, bounty, etc, and the mapping of the accepted requests on top of the discovered IoT resources. It also allows for checking whether the consumers who initiated the `NoD` requests have suff cient fund to pay for their received service.

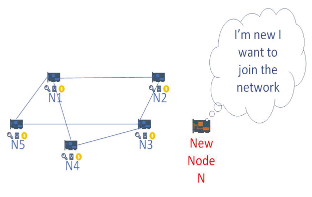
c) Service delivery and integrity: To provide a monitoring mechanism and to ensure service delivery and integrity, a set of monitor devices, denoted by $\mathcal{K}$ and selected during the 2nd phase above, will be appointed to serve as monitors. Their job is to monitor the devices (or providers) that are committed to offer their service to a submitted request by monitoring their uptime, CPU usage, disk usage, network usage, etc. They also act as a fallback mechanism to remove committed devices that fail to deliver and replace them with other nodes. Once failed devices are detected, the monitors intervene by f agging the violating node and issuing a replacement request throughout the network. In `IoTShare`, these monitors act as a f rst-resort for node replacement as soon as service violation/failure is detected, given that they meet the request requirements. This is called *fast recovery response* in our protocol. If the monitors cannot meet the request requirements and thus had to f nd and rely on other nodes in the network to recover from the failure, we refer to this as *slow recovery response*. The f nal scenario happens when no nodes (providers) in the network could be found to fulf ll the request. In this case, the request is dropped, and service delivery failure is recorded and reputation scores of those nodes responsible for such a failure will be reduced accordingly, so that their likelihood of being selected for future requests will be updated. Upon the completion of service (fulf lling the request's required service), all monitors will vote whether to add these providers to the blockchains, BC, based on collected performance metrics and service agreement expectations.

d) Trust and reputation building: `IoTShare` uses and relies on a reputation system that assigns a score to each node to ensure high quality and on time delivery of service. Upon completion of a request, monitors report whether the provider nodes met their service agreements, and assign a score to each provider to ref ect its service quality. The reputation scores of those providers that do not meet their agreed upon service requirements or fail to deliver their service are reset to zero. Monitor nodes create and sign an entry containing these scoring information to be added to the blockchains. The reputation system plays two key roles in `IoTShare`. It ensures high service quality by allowing the selection of reliable and trustworthy nodes, and improves system security by increasing robustness against malicious behavior. `IoTShare`'s adoption of blockchains technology for signing, updating and keeping track of reputation scores of nodes in the network allows consumers to select only those nodes with high reputation by specifying in their request the minimum score a node needs to have to be able to participate in the request. This ensures that high service quality is maintained. Blockchains allows for transparent access to scores and makes it too diff cult for malicious nodes to manipulate and change the scores, increasing system security robustness.

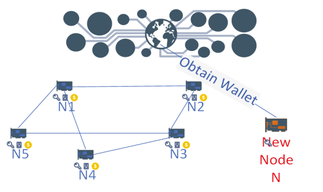
e) Rewarding and payment: `IoTShare` supports billing and reputation management, which is mainly done through BC and the miner nodes. Before a consumer node issues a request, it has to sign the request with its key using its wallet, W, private key (more on this will be discussed later). The request is propagated through the network using peer-2-peer communications to the provider nodes. Provider nodes verify funds availability and request validity using the signature and public key combination. Also, an entry to be added to the BC has to be signed by consumer node and it also has to have monitor nodes signatures for their voting round results. The f rst is used to transfer funds from the consumer node to provider nodes, while the second is used to update the reputation of the provider nodes and to allow a percentage of the funds to go to the monitor nodes. Miners, using proof-of-stake, determine a winner that is appointed to add a new block to the BC, which reduces the computational entry requirement to be a miner in the network.

f) System security and robustness: Every request submitted to the network is signed by the consumer node; i.e., the issuer. Since other nodes in the network can verify the signature using the public key, which is associated with the node's wallet and the message, no node can change the contents of the messages sent in the network. Blockchains entries contain request information, list of accepting nodes, list of accepting monitors, and the mining node. Miners will have to verify every entry for its signature validity, and if an invalid signature is found, then the entry is dropped. When requests are successfully mapped, consumers sign the request, and nodes accepting to serve as providers sign it too, so as to be known which addresses/nodes to be rewarded after service completion. Monitors sign also with voting information as well as with reputation score updates. Also, when a block is found, providers have to verify the signatures attached with the message for validity or else the block is dropped.

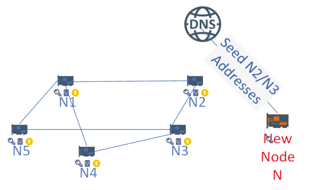
To account for IoT devices' limited storage capacity, `IoTShare` uses compressed Blockchains, where at every period chosen by the protocol, all balances in the network are summed up into states. These states act as a new Block that has to be verified by the miners. Every node has to sign its wallet stating its balance. Miners will verify the signatures and so will nodes that receive the generated block. This in turn minimizes the size of the blockchains and ensures better scalability for our protocol. The number of monitors at each request can be adjusted for to provide better security and robustness and to lower fees associated with each request. Consumers can choose to have no or some monitors to ensure better reliability and service delivery expectations.



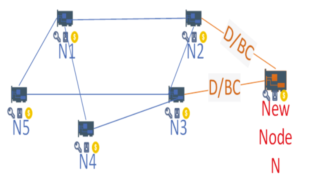
(a) Step 1: A new node N joins the network



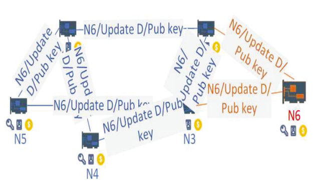
(b) Step 2: node N obtains private and public key pair



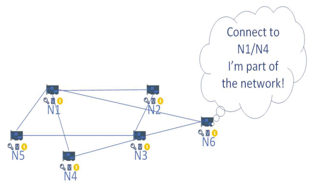
(c) Step 3: Node N queries DNS for addresses of a list of joined nodes (e.g., N2 and N3)



(d) Step 4: Node N obtains `Directory` and BC files



(e) Step 5: Node N assigns itself an ID and propagates its info (e.g., updated `Directory`) to the network



(f) Step 6: Node N picks its neighbors and becomes part of the network

Fig. 2. Illustration of the main steps of the Initialization phase

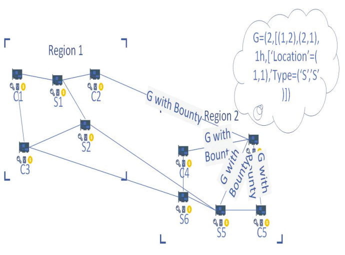
B. `IoTShare` *phases*

`IoTShare`'s operations can be described through three main phases: 'Initialization, 'Resource Reservation' and 'Mining and Fund Transfer'. Initialization phase deals with the joining and registration of newly arrived devices to the network. Resource Reservation phase deals with the mapping of requests, including finding and approving providers, monitors and miners. Fund Transfer phase is the phase where funds are transferred from the consumer to the providers once a request is fulfilled. Throughout, we assume that neighboring nodes are nodes that are connected to the same access point.

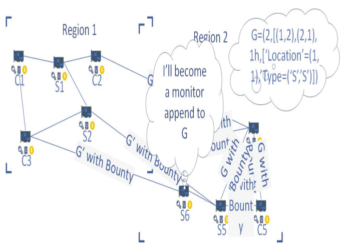
1) Initialization Phase: A new node joining the network has to first obtain W, the Private/Public Key pair that will serve as its wallet address, and BC and `Directory`, the blockchains and device directory files currently being used by the network. The node, for instance, can download and run a software that allows for the creation of a unique Private/Public Key pair. The new node needs to connect to a DNS to obtain a list of nodes in the network from which it can request `Directory` and BC. There could be multiple different provider and miner nodes in the network that can play the role of a DNS, with each node having a different seed that points to a different set of nodes in the network. New nodes can limit themselves with a shortened version of `Directory` that only contains a handful of highly reputed nodes in each region. Also, connecting to neighboring nodes or any node in the network allows new nodes to obtain `Directory`, which will contain updated information about the different resources as well as the nodes in the network.

Once BC is obtained, new nodes can verify the integrity of all transactions throughout the network by verifying the signatures and public keys associated with each entry in BC. Since these transactions are computationally intensive, any non-miner node opts for downloading a light version of BC, which is a blockchains with a shortened length that includes a limited set of past mined blocks, plus a current state of all nodes' balances in the network. If a new node desires to become a provider, it advertises its resources to neighboring nodes, and using gossip protocol [20], all nodes in the network add the new nodes' information to their local `Directory`. The value of R (the node's reputation) is set to 0, and its associated W address is included in `Directory`. A new node can now listen to a specific port for upcoming transactions and for any new requests submitted to the network. These initialization steps are depicted in Fig. 2.

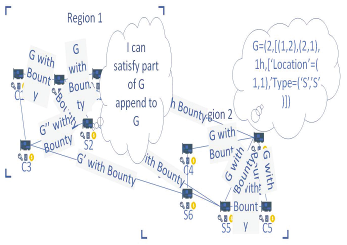
2) Resource Reservation Phase: A consumer node first starts by creating an `NoD` request, G, and prior to submitting it to the network, it must sign the request to enable provider nodes to verify wallet address ownership and funds availability. Although having a full version of `Directory` offers a full view of the network, which can help request specific nodes, to limit the resource reservation overheard, a consumer node opts for a lighter version of `Directory` and limits itself to connecting to a handful of nodes only. Consumer nodes communicate their requests with other nodes using the gossip protocol [20]. Provider and monitor nodes are selected and decided on a first-come first-serve approach basis, with nodes replying to the request first get to take the role. As requests propagate through the network, nodes must adhere to the following; this is also illustrated via Fig. 3.



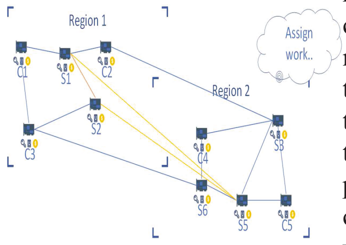
(a) Step 1: S3 creates and formats its request G and sends it to its neighbors


Region 1
Region 2
C1
S1
C2
S2
C3
I'll become a monitor append to G
G=(2,[(1,2),(2,1), 1h,['Location'=(1, 1),'Type=('S','S')])
G' with Bounty
G' with Bounty
G with Bounty
G with Bounty
C4
S6
S5
C5

(b) Step 2(a): Monitors append themselves to the request and forward it to their neighbors



(c) Step 2(b): Providers append themselves to the request and forward it to their neighbors



(d) Step 3: Communication between different nodes is established and S3 assigns work

Fig. 3. Resource reservation illustration: A network with two resource types: sensing (S) or computing (C). S3 is a consumer node with a request G.

- Provider nodes may accept to serve in a number of requests, depending on their locations, resource/time availability, and/or whether they can meet the constraints and/or bounty associated with the request. Nodes could also choose to serve as monitor nodes instead of provider nodes. After deciding (to serve either as monitors or as providers), nodes must append their acceptance information to the request before propagating and passing the request further to the neighboring nodes. Note that as requests pass through the nodes, first nodes to receive the request are more likely to accept it, making it more likely for closer nodes to accept neighboring nodes' requests. This promotes realtime resource reservation with minimum delay.
- The process continues until the request is fully accepted or rejected, where fully accepted means that a provider node has been found for each node of the request. Forever looping can be avoided by limiting the number of hops traversed until the request is fully mapped to some threshold, H, or by using a timeout threshold, T_{max}; i.e., if the request is not fully accepted within a certain time period, it is rejected.
- Once the request is accepted entirely (all of its requested nodes are found), then provider nodes collaborate to deliver the agreed upon service. Note that all provider nodes accepted to participate in a request know about all other (provider and monitor) nodes' information because such an information is appended in the request as the request propagates through. Note that if a request is not accepted, then the requesting consumer node has to create a new request with lower constraints and/or bounty in an effort to find a mapping to the request.

a) Monitoring node functions and selection process: Each accepted request is associated with a number of provider nodes as well as a minimum number, M, of monitoring nodes, as specified by the request. Monitoring nodes are responsible for watching over provider nodes to ensure that they are delivering their service as agreed upon, and do so by monitoring the providers' uptime, CPU usage, disk usage, network usage, etc. This serves two key functions: it serves as backup during node failure recovery, and it helps in updating the reputation and trust scores of the different nodes by scoring providers upon service completion. When committed providers fail to deliver their agreed upon service, monitors, upon detecting such failure, flag the failing nodes to the rest of the network, and issue a node replacement request to replace the failing providers. When no replacement is found, these monitors play the role of immediate node replacement provided that they meet the request requirements. Upon service completion, all monitors initiate a voting process to give and update the reputation scores of the providers based on their received service quality, with the scores of the provider nodes that fail to deliver being set to zero, so that their likelihood of being selected in future requests is reduced.

Nodes can choose to participate as providers or as monitors. As a request propagates through the network, the request is considered to be accepted when a minimum number, M, of nodes are chosen to serve as monitors. To limit maliciousness of possible monitoring nodes, the consumer node also decides on the selection of some of the nodes to serve as monitors, while the other monitors are chosen from random nodes. Also, if nodes were chosen using gossip protocol and adopting the lighter directory file method, monitor nodes append their information to the request as it propagates and the first M nodes to identify themselves as monitors are chosen. Monitor nodes monitor provider nodes as well as other monitor nodes to ensure service agreements are met and to overcome maliciousness.

Their job is to monitor the devices (or providers) that are committed to offer their service to a submitted request by monitoring their uptime, CPU usage, disk usage, network usage, etc. They also act as a fallback mechanism to remove committed devices that fail to deliver and replace them with other nodes. Once failed devices are detected, the monitors intervene by flagging the violating node and issuing a replacement request throughout the network. In `IoTShare`, these monitors act as a first-resort for node replacement as soon as service violation/failure is detected, given that they meet the request requirements. This is called *fast recovery response* in our protocol. If the monitors cannot meet the request requirements and thus had to find and rely on other nodes in the network to recover from the failure, we refer to this as *slow recovery response*. The final scenario happens when no nodes (providers) in the network could be found to fulfill the request. In this case, the request is dropped, and service delivery failure is recorded and reputation scores of those nodes responsible for such a failure will be reduced accordingly, so that their likelihood of being selected for future requests will be updated. Upon the completion of service (fulfilling the request's required service), all monitors will vote whether to add these providers to the blockchains, BC, based on collected performance metrics and service agreement expectations.

b) Node reputation update and failure recovery: Monitor nodes have the ability to raise a flag indicating a misbehavior from a specific provider node. In this case, a voting round including all monitor nodes is done, during which it will be decided whether the said provider node violated the service agreement or not. If the number of votes is higher than a threshold, V_{min}, then the node is flagged as misbehaved and its misconduct is broadcasted throughout the network. Reputation is set to 0 for the misbehaving node and the accepted request is re-advertised to neighboring nodes. If a provider node went down during task performance, its status in the directory is changed to inactive by monitor nodes and is broadcasted throughout the network. Monitor nodes get to accept the resources first because of their proximity from the misbehaving node. This creates another incentive for monitor nodes to keep monitoring provider nodes to detect failures. Once a request is over and if no flags were raised, monitor nodes get to vote to decide whether provider nodes met the service agreement of the request. If votes exceed V_{min}, a new entry that includes consumer node, provider nodes, monitor nodes, and the request G is broadcasted to be added to the BC. Monitors receive a Monitor Bounty, Mb, as a reward for their monitoring service, which is a percentage of the bounty associated with the request. This provides an incentive to do the monitoring job. The provider nodes' reputations are increased for their successful fulfillment of the served request.

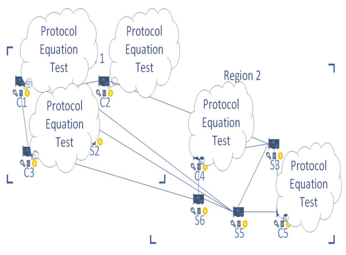
3) Mining and Fund Transfer Phase: Miner nodes listen to broadcasts sent by consumer nodes, provider nodes and monitor nodes at all times. They add transactions that include fulfilled requests, reputation changes to their backlog once received. As transactions accumulate, miner nodes verify each transaction using the public key and signature associated with each transaction. At every mining period chosen by the protocol, a miner node is selected at random to add a block to the blockchains, BC. The probability of a miner node being selected is related to how much stake it has in the system (Total funds). Miner nodes with more stake will have higher probability of winning the mining task. To be a miner winner, a node must satisfy:

$$Hash(PrevHash, WalletAddr, Content, Time) \leq \frac{Balance(WalletAddr)}{TotalBalance} \times Difficulty \quad (1)$$

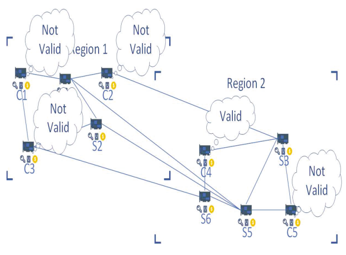
where:

- $PreviousHash$: Hash generated for previous block.
- $Content$: Transaction(s) to be added to the block.
- $WalletAddr$: Public key associated with the miner node.
- $Time$: Period selected to produce a miner node winner.
- $Balance(WalletAddr)$: Balance of the miner node.
- $TotalBalance$: Total fund available for all nodes.
- $Difficulty$: Current difficulty level set for network.

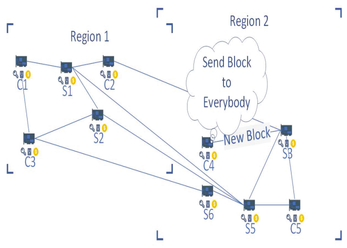
a) Miner selection: Given a miner's input parameters (i.e., $PrevHash, WalletAddr, Content, Time$), the hash function produces a value that is used to determine whether the miner wins. Here, for a given $Difficulty$, the higher



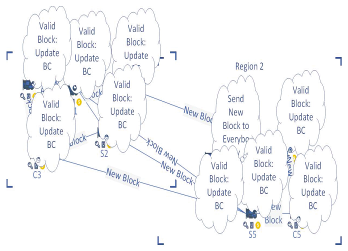
(a) Step 1: Any node wanting to be a miner checks the inequality.



(b) Step 2: Only node C4 satisfies the inequality and wins the mining.



(c) Step 3: The winning miner sends the new block to its neighbors.



(d) Step 4: Nodes verify inequality satisfaction and block validity. When so, new block is added to BC.

Fig. 4. Single-miner winner illustration: An IoT network with two types of resources: sensing (S) or computing (C).

the miner's stake (i.e., balance $Balance(WalletAddr)$), the greater the chance of being selected as a miner. The adjustable parameter, $Difficulty$, serves as a way for ensuring that only one winner miner is likely to be selected. That is, the smaller the value/level of the chosen $Difficulty$ parameter is, the tighter Inequality (1) is, and hence, the less likely the inequality is to be met. Every miner computes this hash function and checks whether the hash value satisfies the inequality, and if it does (and hence the miner wins this mining period), it advertises the new block to the rest of the network so the block can be added. This is illustrated in Fig. 4. In case no miner satisfies this inequality and no nodes in the network receive a valid block by next the mining period (period selected by the protocol to produce a miner node winner), all nodes in the network decrease the value of $Difficulty$ to ensure that at the next mining period, the likelihood of finding a miner increases. If two or more miners satisfy the inequality and broadcast the new block to be added, the block is dropped by all nodes and the $Difficulty$ is increased, so as again to reduce the likelihood of having multiple miners satisfying the inequality by the next mining period.

b) Mining reward: New mined blocks are unconfirmed until L_{min} blocks have been mined. Once the number of blocks that have been mined reaches L_{min}, all the previous transactions in current block are said to be confirmed and the winner miner receives a reward, R (Miner Reward)). This gives an incentive for miner nodes to participate in the miner selection process. If one of the miner nodes attempted to create fake entries in the BC and broadcast it to the rest of the network, other nodes in the network would drop it if it does not satisfy the inequality above.

IV. Performance Evaluation of `IoTShare`

In this section, we present and analyze the performance results of the proposed protocol, `IoTShare`, using simulation.

A. Simulation Method and Setup

1) Network setup: We consider a network of N nodes/IoT devices placed randomly in a city. The network is modelled as a graph whose vertex set is the set of all N nodes and the edge set consists of random connections among the nodes in the system. We assume that the city is split into L regions, indexed $0, 1, ..., L-1$, and the devices are distributed randomly within the different regions. In our simulation, the number of neighbors each node is directly connected to is selected uniformly between $Neigh_{min}$ and $Neigh_{max}$. Only fully connected graphs, where each node can be reached from any other node, are simulated. Each node is randomly assigned a resource type, which can be either of sensing or computing type. At any time slot, a node can be in one of the following states: *idle, provider, monitor, miner or failed node*. Initially, a node starts in idle state, and as networks on-demand requests arrive, its state changes accordingly; e.g., when a node accepts to serve in a request, it changes its state to provider; when it accepts to serve as a miner, it changes its state to a miner, etc.

2) `NoD` *requests:* We consider a time-slotted system, with the number of requests that arrive at each time step is Poisson distributed with mean r, and the duration of each accepted request follows a Bernoulli process with parameter q. Every `NoD` request comes in the form of 5-tuple G=(V, $\mathcal{E}$, D, $\mathcal{C}$, B), with V specifying the request size (i.e., the number of requested nodes), $\mathcal{E}$ the set of connections/edges between the requested nodes, D specifying the request duration, $\mathcal{C}$= $\{(Loc_1, Type_1, Cap_1), \ldots, (Loc_V, Type_V, Cap_V)\}$ specifying the (location, resource type, resource capacity) of each requested node, and B specifying the bounty associated with the request. Upon its arrival, an `NoD` request is propagated across the network, and depending on the request's requirements and the devices' availabilities and willingness to serve (e.g., the device has the requested type of resource and the requested resource capacity, the device's minimum acceptable bounty is met, etc.), it can be either accepted or denied.

3) Mining: At every time period, a new miner is selected as described in Section III-B3. When a request is successfully accepted, it is propagated throughout the network and added to the miners' pending ledger. Every winning miner picks from the pending requests based on the highest bounty associated with the pending requests. We assume that nodes are faulty (whether intentionally or unintentionally), in that a provider accepting to serve a request may, with some probability, fail to deliver its service, thereby causing service disruption and violation of service agreements. `IoTShare` contains a mechanism that allows recovery from such failures by promptly f nding other nodes that can fulf l and replace the failing nodes. For convenience and completeness, we summarize and provide all of the notations and variables used in this work in Table I.

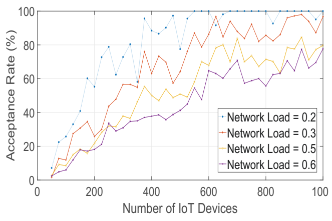
4) Parameter setting: Each node maintains the structure `Directory`, which contains ID, resource type, resource capacity, location, time availability, and minimum required bounty of each participatory device in the network. In our evaluation, the ID assigned to each node ranges from 1 to V, the resource type is set to either 0 for sensing or 1 for computation, and each node starts with zero balance, and the balance increases as the node participates in serving requests and receives more rewards. We distribute nodes over a 2D plane and assign each of them a random x,y coordinate to represent its location. The 2D plane is divided into $L = 9$ regions, where each node is placed randomly in one of the regions. Each node is connected to other neighboring nodes, with the number of neighbors selected uniformly between $Neigh_{min} = 15$ and $Neigh_{max} = 25$.

In our simulation, for each arrived request, the request size, V, is varied between 5 and 20, the request duration, D, is drawn from a Bernoulli process with parameter q i.e., average duration (in number of time slots) equals $1/q$), the resource type of the request is set to 0 (sensing) or 1 (computing) with equal probability, the locations of requested nodes are selected randomly, and the request bounty, B, is selected uniformly between 100 and 1000. Each node has a minimum bounty, B_{min}, below which a request is denied. In our simulation, B_{min} is also drawn uniformly between 100 and 1000.

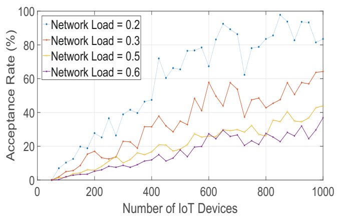
We def ne the *network load* as r/q, where again r is the parameter of the Poisson distribution representing the number of requests that arrive per time slot. r/q here represents the average number of requests that would have been present in the network at a time slot had no arrived requests been denied to the network. In other words, the average number of requests that are actually present in the network equals the "network load" times the "accepted rate" of arrived requests. In our simulation, we varied the network load between 0.2 and 0.6. Finally, the size of mining period, τ_{mining}, (one block is added to the blockchains every mining period) is set to 3 time slots, and the number of monitors, M, is set also to 3 in our

TABLE I
`IoTShare` PARAMETERS

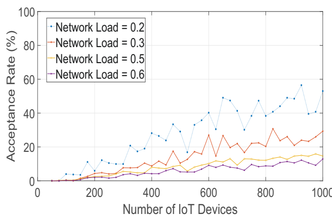
G	$\triangleq$	5-tuple request form
D	$\triangleq$	Request duration
$\mathcal{C}$	$\triangleq$	(locations, types, capacities) of requested nodes
B	$\triangleq$	Request bounty
N	$\triangleq$	Total number of nodes/IoT devices in the network
`Directory`	$\triangleq$	Device directory
L	$\triangleq$	Number of city regions
BC	$\triangleq$	Blockchains used by the protocol
W	$\triangleq$	IoT device wallet
BC'	$\triangleq$	Shortened (lightweight) blockchains
H	$\triangleq$	Request hop limit value
T_{max}	$\triangleq$	Timeout duration for request
M	$\triangleq$	Number of monitors per request
V_{min}	$\triangleq$	Monitor minimum voting threshold
R	$\triangleq$	Miner reward value
B_{min}	$\triangleq$	Min bounty for a provider to accept a request
τ_{mining}	$\triangleq$	Miner selection and block addition period
$Neigh_{min}$	$\triangleq$	Min number of neighboring nodes per node
$Neigh_{max}$	$\triangleq$	Max number of neighboring nodes per node
r	$\triangleq$	Average number of requests arrive per time slot
q	$\triangleq$	Bernoulli process parameter; $1/q$ is average duration (in time slots) of accepted request



(a) Request size = 5

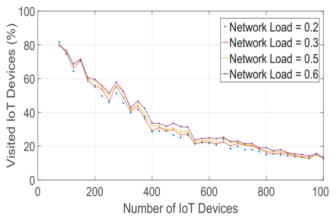


(b) Request size = 10

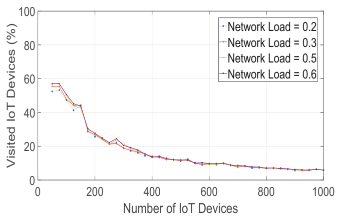


(c) Request size = 20

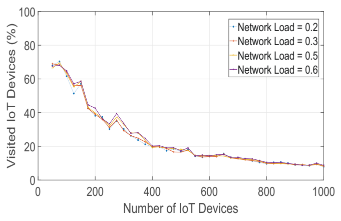
Fig. 5. Acceptance rate performance



(a) Request size = 5



(b) Request size = 10



(c) Request size = 20

Fig. 6. Fraction of devices visited prior to request mapping.

simulations.

B. Performance Metrics

To assess the effectiveness of `IoTShare`, we measure and evaluate the following performance metrics:

- **Acceptance Rate**: The rate at which requests are accepted into the network. It is calculated by dividing the number of accepted requests by the total number of requests submitted to the network.
- **Visited IoT Devices**: The percentage/fraction of IoT devices in the system that are visited before a request's requirements are satisfied and the request is accepted. It is averaged over all accepted requests.
- **Blockchains Size**: The number of transactions added per block. It is averaged over all blocks added to the blockchains.
- **Recovery Rate**: The fraction of accepted requests that are recovered successfully from device failures. It is calculated by counting the number of successfully recovered requests and dividing it by the total number of failed requests.
- **Mining Frequency**: It is the fraction of mining times a miner has been selected to serve as a miner. Each miner has its own 'Mining Frequency' value. It is calculated as the number of times a miner has been selected as a miner divided by the total number of miner selections or mining periods. This reflects the robustness of `IoTShare` to the 51% Attack problem [15].

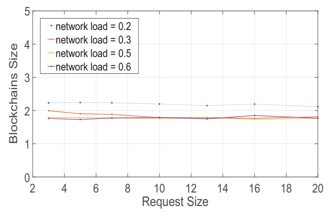
C. Performance Analysis

Our performance analysis of `IoTShare` is classified into two categories, scalability, fault-tolerance, and robustness, each of which is discussed in a separate section.

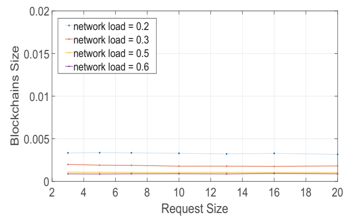
1) Scalability: We study the performance behavior of `IoTShare` to assess its ability to scale with the network size (number of IoT devices) by investigating the impact of the network size on three metrics: Acceptance Rate, Visited IoT Devices, and Blockchains Size. In Figure 5, we present the acceptance rate while varying the number of IoT devices under different network loads. Each graph represents a different request size, 5, 10 and 20 from top to bottom, where again the request size is the number of IoT devices requested. As expected, we observe from the figures that the acceptance rate decreases as the network load and/or the request size increases. This is because as the network load and/or request size increases, more nodes in the network become committed to requests, making it harder to find nodes that satisfy the new requests' requirements. We also note that as the network size grows, the acceptance rate increases, merely because more nodes in the network implies higher chances of meeting new requests' requirements and hence higher acceptance rates.

In Figure 6, we present the number of nodes (in percentage) that are visited prior to fulfilling a request under different network loads. Each graph corresponds to a different request sizes, 5, 10 and 20 from top to bottom. We observe that as the network grows larger, lesser percentage of the nodes need to be visited to fulfill a request. This is because as the number of nodes increases, the likelihood of finding nodes that meet the request increases as well, which decreases the percentage of nodes that need to be visited before satisfying a request. The figure also shows that the network load does not affect the percentage of nodes visited prior to fulfilling a request. The reasoning behind this is that both committed and non-committed nodes have to be visited to check for their availability to serve in and accept the request, so a higher percentage of committed nodes does not impact the number of nodes that need to be visited to fulfill a request. We also observe that as the request size grows larger, more nodes need to be visited to fulfill a request. This is because as the request size gets bigger, there are more nodes that need to be picked to satisfy a request, which in turn increases the overall number of visited nodes.

In Figure 7(a), we present the blockchains size/overhead incurred per mining period, i.e., per added block, as a function of the request size under different network loads. The figure

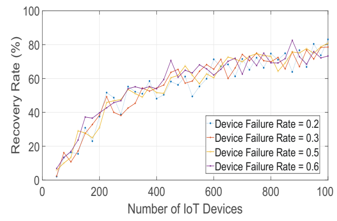


(a) Per mining period

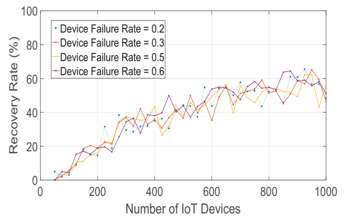


(b) Per mining period per request

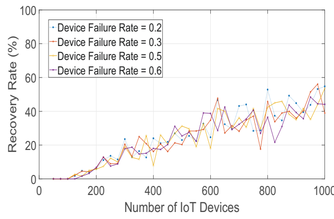
Fig. 7. Blockchains size overhead (a) per mining period and (b) per mining period per arrived request



(a) Request size = 5

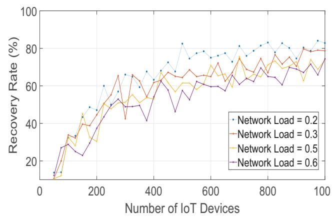


(b) Request size = 10

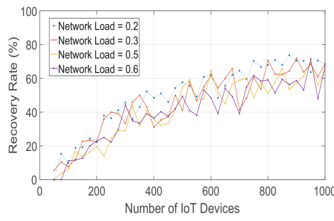


(c) Request size = 20

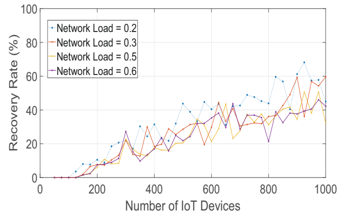
Fig. 8. Failure recovery rate performance: network load = 0.5



(a) Request size = 5



(b) Request size = 10



(c) Request size = 20

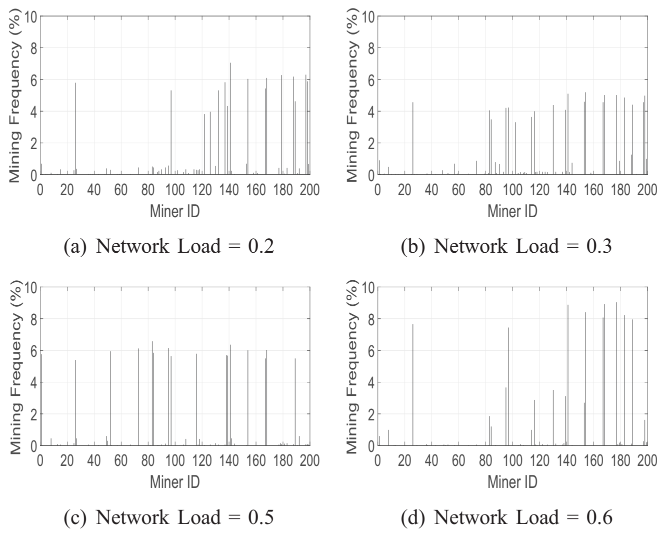
Fig. 9. Failure recovery rate performance: device failure rate = 0.2

shows that the request size does not have an effect on the blockchains size. When the request size increases, it means that more nodes, on average, are getting selected per request, but without incurring much blockchains overhead. We also note that as the network load increases, the overhead decreases slightly. As the network load increases, more requests are coming into the network, dropping the rate of accepted requests, which in turn causes the average number of transactions per time slot to drop slightly. In Figure 7(b), we present the blockchains size incurred per arrived request (we divide the blockchains size by the number of arrived requests per mining period). We also notice that the network load and request size have minimal effect on the blockchains overhead in terms of required blockchains size. To summarize, `IoTShare` scales very well in terms of blockchains overhead under different network loads and request sizes.

2) Fault tolerance: We now study the robustness of `IoTShare` against node failures. For this, we consider that nodes can fail before or after accepting a request, and assess how well `IoTShare` recovers from such failures by measuring the recovery rate of the protocol under different failure rates, request sizes, and network loads. The recovery rate metric is already defined in Section IV-B.

Figures 8 and 9 show the behavior of the recovery rate as a function of the number of IoT devices under different network loads (0.2, 0.3, 0.5 and 0.6), request sizes (5, 10 and 20), and node failure rates (probability of device failure is set to 0.2, 0.3, 0.5, and 0.6). We make the following three observations. First, observe that as the number of IoT devices in the system increases, the recovery rate increases, regardless of the network load, the request size, and the node failure rate. This is because the higher the number of nodes, the greater the likelihood of finding nodes that satisfy the failed nodes' requirements, thus increasing the overall recovery rate. For reasonable network sizes (e.g., 1000), the recovery rate ranges from 50 to 80%, depending on the network load, node failure and request size. Second, note that the device failure rate and the network load has little effect on the recovery rate. The reasoning behind this is that, for e.g., as the device failure rate increases, `IoTShare` is still able to recover from failures that happen to different nodes in the network. Since the load is constant, the likelihood of finding a node that satisfies the failed network requirement is the same. Third, note that as the request size increases, the recovery rate declines slightly, and this is regardless of the node failure rate and/or the network load. As the request size gets bigger, more nodes are committed, and hence, it becomes difficult to find replacement for failed nodes, but the effect is minimal as we only see a slight drop in the recovery rate. This is due to the fact that there are still unoccupied nodes throughout the network that can be picked as a replacement for the failed nodes. To summarize, our results show that `IoTShare` is robust against faulty nodes, by being able to achieve high recovery rates under reasonable network and request sizes.

3) Robustness: We now assess `IoTShare`'s robustness against the 51% attack inherent to the blockchains' mining mechanism [15]. For this, we measure and show in Figure 10 the mining frequency of each miner under four different network loads, where the mining frequency is calculated as the number of times a miner has been selected as a miner divided by the total number of miner selections or mining periods. In this experiment, the number of miners is set to 200; i.e., only 200 IoT devices among all devices may serve as miners. The figure shows that, regardless of the network



(a) Network Load = 0.2 (b) Network Load = 0.3

(c) Network Load = 0.5 (d) Network Load = 0.6

Fig. 10. Miner selection frequency: number of miners = 200 (IDs: 1 to 200).

load, no single miner is selected more than 7% (except in the case of network load of 0.6, some received 9%), and more importantly, no miner is selected overwhelmingly more than the other miners. This shows that `IoTShare`'s blockchains-enabled mechanism is robust to the 51% attack.

V. Security Considerations

`IoTShare` adopts mechanisms to address some security challenges in addition to enabling and easing the mapping, reservation, and sharing of IoT resources on-demand and in a distributed manner. In this section, we talk about key security threats and attacks and discuss whether and how `IoTShare` addresses them.

a) Sybil attacks: One of such attacks is Sybil attacks [21], which blockchains-based protocols in general suffer from. In `IoTShare`, these attacks can come from consumers and/or providers that maliciously fake their roles. Malicious consumers may generate lots of fake requests to bog down the network, and provider nodes can initiate Sybil attacks by promising to serve in requests but without delivering service. There are three approaches `IoTShare` adopts to mitigate the impact of Sybil attacks. The f rst one is by relying on PoS approach when deciding on and selecting providers to serve on requests. Providers with high stake values are not encouraged to and have no interest in launching such attacks. `IoTShare` requires a minimum stake threshold for users to service as providers. The second approach is through reputation scoring, which can help in f ltering out malicious users in the long term, thereby getting rid of users that have no good intention to serve. The third approach is to have `IoTShare` check for balance and make sure that the consumer's wallet has suff cient fund prior to allocating resources to its request, thereby discouraging users from playing fake consumer roles.

b) Bribe attacks: Bribe attacks are another type of attacks that can be launched by monitor nodes whose main job is to monitor committed providers and give reputation scores via voting based on the quality of providers' delivered service. Monitor nodes can collude and falsify voting and reputation score decisions. Such attacks can also be mitigated through PoS and by imposing minimum stake in order for users to serve as monitors, thus discouraging them from colluding with the aim to fake the system.

c) Long-Range attacks: A third type of attacks inherent to the PoS approach that `IoTShare` tackles is Long-Range attacks. These attacks occur when an attacker goes back and forks the original chain to create a branch with different blocks, with the newly created branch overtaking the main blockchain by becoming longer than the main chain. PoS protocols are more vulnerable than PoW protocols, because the former type does not require computational effort to generate previous added blocks until the branch outpaces the main chain [22]. Generally, these attacks are classif ed in three main categories, Simple, Posterior Corruption, and Stake Bleeding, with varying complexity and assumptions (e.g., ability to forge timestamps vs. not, ability to collude with other miners/monitors vs. not) [22]. Simple attacks exploit trust-based PoS implementation in which timestamps are not checked by nodes at every block addition, and hence, each node can validate the new blocks. The other two categories execute more complex attacks. There have been countermeasure techniques proposed in the literature to address long-range attacks, including longest chain rule, key-evolving cryptography [23] and plenitude rule [24] among others, and `IoTShare` can use a combination of these to protect against such attacks. More information on these attacks as well as on other PoS attack types can be found in [22].

d) Nothing-At-Stake attacks: Like most PoS protocols, `IoTShare` relies on longest chain policy to resolve forking, which makes sure that one fork/branch eventually overtakes the other branches. However, this policy may suffer from the nothing-at-stake attacks, which refer to the scenario when nothing stops/discourages nodes from mining conf icting blocks (to be added to multiple different forks) as this does not risk their stake. This attack leads to delaying the time for reaching a consensus as well as increasing the number of branches in the blockchain and potentially allowing double-spending. In `IoTShare`, we propose to combine Ouroboros [25] and Slasher [26] to mitigate this attack. Ouroboros incorporates a rewarding strategy to discourage nodes from mining and adding blocks to multiple different branches, whereas Slasher proposes that each miner makes a deposit prior to each mined block, and the deposit is locked for some period of time. The deposit is lost if a node signs and adds blocks to different branches with the same length.

VI. Open Research Challenges

`IoTShare` is a fully distributed protocol designed to be specif cally suitable for smart cities, by easing and enabling the creation and deployment of multiple networks-on-demand instances on top of IoT devices to support various smart city applications and services, including surveillance applications, traff c control applications, emergency relief management applications, law enforcement applications, and many others. Our greater vision for `IoTShare` is to move from simulation based deployment and assessment to a real platform implementation and evaluation. In order to reach this goal, few challenges remain to be addressed.

a) Compatibility: It is assumed that this technology could be deployed on all devices without concerning ourselves with the possibility that different devices may need different types of API/programming language/code to be able to be integrated into the system. One possible solution is to use a universal API middle man translator (broker) that all systems could talk to using a web REST API. So, one research task would be to look into how to best use REST API.

b) Power Conservation: Power/energy preservation has not been accounted for in this research. `IoTShare` as of right now involves a lot of data transfers and communications among the IoT devices (e.g. using gossip). Since IoT devices typically have limited energy resources, power consumption cloud be a limiting factor. A future research item would be to focus on optimizing data transfers/IoT utilization with energy consumption awareness in mind.

c) Incentive Mechanisms: `IoTShare` assumes that IoT devices will be encouraged to join the network and be part of networks-on-demand instances. Although the bounty/reward given to a device as a reward to its service can serve as an incentive, we believe more carefully thought out schemes need to be investigated. For instance, resource availability would be a great barrier for these devices to join and participate, and if and when provided with solutions that can alleviate the need for such resources, these devices will be more inclined to join. One potential technology that can serve this purpose is edge cloud computing, which is already shown to allow for device resource (computation, storage, network, etc.) off oading, providing therefore great incentives to join as they no longer need to use their local resources, at least fully. However, more needs to be done when it comes to integrating `IoTShare` with edge cloud technology.

d) Blockchains Storage Limitation: While `IoTShare` allows and supports the use of a lighter version of blockchains, more can be done in this regard. Not all IoT devices have the storage capabilities to store very large blockchains data, and hence, a future research task could be to look into optimizing blockchains structure and creating an architecture that would allow IoT devices to keep the integrity of the blockchains while also limiting storage overhead.

e) Patching/System Updates Concerns: One task that could be looked at in great depth is to design eff cient mechanisms that would be used to push updates to the network and to endpoints in the network. For instance, once `IoTShare` is deployed, should we use a central code version that is monitored by the owners of the network to allow for system updates/patches to be deployed, or should we leave updates to be independent so each provider could deploy any code into the network as long as it is able to communicate with the rest of the network? These questions need be investigated.

f) Monitoring Power: In `IoTShare`, monitors are the nodes that decide whether a provider delivers its agreed upon service or not. But if the monitors (intentionally or unintentionally) decide to reject providers' work, then the system as a whole would fail. A future research task would be to investigate eff cient mechanisms that can be used to choose monitors and to ensure that monitors do their job correctly.

g) Monitoring System Meta Data: One of the monitors' job is to ensure that providers perform their task as agreed upon, but not much emphasis is put into how that is done. It is assumed that the monitors would have access to some of the meta data/health checks of the end points/providers. A future research task would then be to design eff cient schemes/architectures that address these challenges.

h) Market Analysis: `IoTShare` relies on a supply/demand system, in that when there are a lot of requests (resulting in increasing the demand for IoT devices to serve as providers), the bounty associated with these requests needs to be adjusted so that devices could be found to fulf l the requests. A future task is to look into mechanisms that can be used to adjust the bounty values to ensure that the system does not break. At which point would the consumers start building/buying their own hardware instead of trying to reserve it from the network? Should there be some kind of bounty capping mechanism to ensure that we do not reach that point? These questions need also be investigated.

VII. Conclusion

In this paper, we presented `IoTShare`, a blockchains-based, distributed IoT resource sharing on-demand protocol. `IoTShare` uses blockchains technology to enable distributed mapping and management of networks on-demand on top of IoT devices. `IoTShare` can be used in smart cities to allow the deployment of networks-on-demand instances on top of IoT devices located within a city. These networks-on-demand instances can enable and support a variety of applications to offer services that can, for example, help with surveillance, emergency management, traff c control, and many other city related matters. We showed through simulations that `IoTShare` scales eff ciently under different system parameters, resilient to faulty nodes, and is robust against the 51% attack.

References

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] B. Hamdaoui, M. Alkalbani, T. Znati, and A. Rayes, "Unleashing the power of participatory iot with blockchains for increased safety and situation awareness of smart cities," *IEEE Network*, pp. 1–8, 2019.

[3] B. Hamdaoui and K. G. Shin, "Characterization and analysis of multi-hop wireless mimo network throughput," in *Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*. ACM, 2007, pp. 120–129.

[4] Y. Zhang, J. Zheng, and H.-H. Chen, *Cognitive radio networks: architectures, protocols, and standards*. CRC press, 2016.

[5] M. Rebato, F. Boccardi, M. Mezzavilla, S. Rangan, and M. Zorzi, "Hybrid spectrum access for mmwave networks," in *2016 mediterranean ad hoc networking workshop (Med-Hoc-Net)*. IEEE, 2016, pp. 1–7.

[6] B. Hamdaoui, B. Khalf , and M. Guizani, "Compressed wideband spectrum sensing: Concept, challenges, and enablers," *IEEE Communications Magazine*, vol. 56, no. 4, pp. 136–141, 2018.

[7] H. Sinky, B. Khalf , B. Hamdaoui, and A. Rayes, "Responsive content-centric delivery in large urban communication networks: A linknyc use-case," *IEEE Transactions on Wireless Communications*, vol. 17, no. 3, pp. 1688–1699, 2017.

[8] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," in *Proceedings of the second edition of the ICN workshop on Information-centric networking*. ACM, 2012, pp. 55–60.

[9] R. Abuhadra and B. Hamdaoui, “Proactive in-network caching for mobile on-demand video streaming,” in *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018, pp. 1–6.
[10] X. Chen, L. Jiao, W. Li, and X. Fu, “Eff cient multi-user computation off oading for mobile-edge cloud computing,” *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2015.
[11] S. Abdelwahab, S. Zhang, A. Greenacre, K. Ovesen, K. Bergman, and B. Hamdaoui, “When clones f ock near the fog,” *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1914–1923, 2018.
[12] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2008.
[13] M. Singh, A. Singh, and S. Kim, “Blockchain: A game changer for securing iot data,” in *Internet of Things (WF-IoT), 2018 IEEE 4th World Forum on*. IEEE, 2018, pp. 51–55.
[14] M. Samaniego and R. Deters, “Blockchain as a service for iot,” in *Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 2016 IEEE International Conference on*. IEEE, 2016, pp. 433–436.
[15] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld, “Proof of activity: Extending bitcoin’s proof of work via proof of stake [extended abstract],” *ACM SIGMETRICS Performance Evaluation Review*, vol. 42, no. 3, pp. 34–37, 2014.
[16] D. Miller, “Blockchain and the internet of things in the industrial sector,” *IT Professional*, vol. 20, no. 3, pp. 15–18, 2018.
[17] S. Huh, S. Cho, and S. Kim, “Managing iot devices using blockchain platform,” in *Advanced Communication Technology (ICACT), 2017 19th International Conference on*. IEEE, 2017, pp. 464–467.
[18] M. Samaniego and R. Deters, “Hosting virtual iot resources on edge-hosts with blockchain,” in *Computer and Information Technology (CIT), 2016 IEEE International Conference on*. IEEE, 2016, pp. 116–119.
[19] G. O. Karame, E. Androulaki, and S. Capkun, “Double-spending fast payments in bitcoin,” in *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, 2012, pp. 906–917.
[20] R. Chandra, V. Ramasubramanian, and K. Birman, “Anonymous gossip: Improving multicast reliability in mobile ad-hoc networks,” in *Distributed Computing Systems, 2001. 21st International Conference on.* IEEE, 2001, pp. 275–283.
[21] J. Douceur, “The sybil attack,” in *Peer-Peer Systems, 2002 International Conference on*. Springer, 2002, pp. 251–260.
[22] E. Deirmentzoglou, G. Papakyriakopoulos, and C. Patsakis, “A survey on long-range attacks for proof of stake protocols,” *IEEE Access*, 2019.
[23] B. David, P. Gaži, A. Kiayias, and A. Russell, “Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain,” in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2018, pp. 66–98.
[24] C. Badertscher, P. Gaži, A. Kiayias, A. Russell, and V. Zikas, “Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability,” in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 913–930.
[25] A. Kiayias, A. Russell, B. David, and R. Oliynykov, “Ouroboros: A provably secure proof-of-stake blockchain protocol,” in *Annual International Cryptology Conference*. Springer, 2017, pp. 357–388.
[26] V. Buterin, “Slasher: A punitive proof-of-stake algorithm,” *Ethereum Blog URL: https://blog. ethereum. org/2014/01/15/slasher-a-punitive-proof-of-stake-algorithm*, 2014.

Bechir Hamdaoui (S’02-M’05-SM’12) is a Professor in the School of EECS at Oregon State University. He received M.S. degrees in both ECE (2002) and CS (2004), and the Ph.D. degree in ECE (2005) all from the University of Wisconsin-Madison. His research interests are in the general areas of computer networks, wireless communication, and computer security. He won several awards, including the ICC 2017 and IWCMC 2017 Best Paper Awards, the 2016 EECS Outstanding Research Award, and the 2009 NSF CAREER Award. He serves/served as an Associate Editor for several journals, including IEEE Transactions on Mobile Computing, IEEE Transactions on Wireless Communications, IEEE Network, and IEEE Transactions on Vehicular Technology. He also chaired/co-chaired many IEEE conference programs/symposia, including the 2017 INFOCOM Demo/Posters program, the 2016 IEEE GLOBECOM Mobile and Wireless Networks symposium, and many others. He served as a Distinguished Lecturer for the IEEE Communication Society for 2016 and 2017. He is a Senior Member of IEEE.

Mohamed Alkalbani received the B.S. and M.S. degrees in ECE from Oregon State University, in 2016 and 2018, respectively, and is currently working for HP as a software engineer. His research interests include computer networks, distributed systems, and blockchain technology.

Ammar Rayes (S’85-M’91-SM’15) is a Distinguished Engineer / Senior Director at Cisco Services Chief Technology and Strategy Off ce working on the Technology Strategy. His research interests include Network Analytics, IoT, Machine Learning and NMS/OSS. He has authored over 100 publications in refereed journals and conferences on advances in software & networking related technologies, 4 Books and over 30 US and International patents. He is the Founding President and board member of the International Society of Service Innovation Professionals www.issip.org, Adjunct Professor at San Jose State University, Editor-in-Chief of Advances of Internet of Things Journal, Editorial Board Member of IEEE Blockchain Newsletter, Transactions on Industrial Networks and Intelligent Systems, Journal of Electronic Research and Application and the European Alliance for Innovation - Industrial Networks and Intelligent Systems. At Cisco, Ammar is the founding chair of Cisco Services Research and Cisco Services Patent Council. He received Cisco Chairman’s Choice Award for IoT Excellent Innovation & Execution. He received his BS and MS Degrees in EE from the University of Illinois at Urbana and his Ph.D. degree in EE from Washington University in St. Louis, Missouri, where he received the Outstanding Graduate Student Award in Telecommunications.

Nizar Zorba (SM’18) is a Professor at the Electrical Engineering department at Qatar University, Doha, Qatar. He has authored f ve international patents and co-authored over 120 papers in peer-reviewed journals and international conferences. Dr. Zorba received the B.Sc. degree in electrical engineering from JUST University, Jordan, in 2002, and the Ph.D. degree in signal processing for communications from UPC Barcelona, Spain, in 2007. He is associate/guest editor for the IEEE Communications Letters, IEEE Access, IEEE Communications Magazine and IEEE Network. Currently, he is the vice-chair of the IEEE ComSoc Communication Systems Integration and Modeling Technical Committee (TC CSIM).