2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom) | 978-1-6654-9425-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/TrustCom56396.2022.00129

A Multidimensional Blockchain Framework For Mobile Internet of Things

Hussein Zangoti[1 2], Alex Pissinou Makki[3], Niki Pissinou[1], Abdur R. Shahid[4], Omar J. Guerra[1*], Joel Rodriguez[1*]
[1]*School of Computing and Information Sciences, Florida International University, Miami, FL, USA*
[2]*Faculty of Computer Science & Information Technology, Jazan University, Jazan, Saudi Arabia*
[3]*College of Engineering, University of California, Berkeley, CA, USA*
[4]*Department of Computer and Information Systems, Robert Morris University, Moon, PA, USA*
[1]{hzang001, pissinou, oguer022, jrodr173}@fiu.edu, [3]alexpissinoumakki@berkeley.edu, [4]shahid@rmu.edu

***Abstract*—The adoption of blockchain in the Internet of Things (IoT) has been increasing due to the various benefits that blockchain brings, such as security and privacy. Current blockchain models for mobile IoT assume there are fixed, powerful edge devices capable of providing global communication to all the nodes in the network. However, due to the mobile nature of IoT or network partitioning problems (NPP), nodes can move out of a cell area and split into smaller independent peer-to-peer subnetworks. Existing blockchain structures either do not support the network partitioning problem or have limitations. This paper introduces a multidimensional, graph-based blockchain structure, that utilizes k-dimensional spatiotemporal space, to address the challenges of applying blockchain in mobile networks with limited resources. Experimental results show that a multidimensional blockchain structure can improve scalability and efficiency as the blockchain grows in size, similar to logarithmic growth, and reduce the longest chain length by more than 99.99% compared to the traditional chain-based blockchain structure.**

***Index Terms*—Multidimensional, Blockchain, IoT, WSN, Cryptography, Binary Search Trees, Network Partitioning, Mobility.**

I. Introduction

Blockchain presented by Nakamoto [1] is a time-stamped append-only log technology that is usually decentralized, immutable and has led to innovative applications in many areas such as finance, healthcare, distributed systems, voting, industry, and real estate. When integrating blockchain into other domains with limited resources, such as mobile IoT, which is the focus of this paper, there are multiple challenges to address. Traditional IoT blockchain systems rely on high network connectivity, which implies they depend on fixed, powerful edge devices capable of continuously providing global communication to all the nodes Karlsson *et al.* [2]. However, considering the mobility nature of IoT, nodes can split into smaller and independent peer-to-peer subnetworks due to the absence of edge devices. Existing blockchain structures either do not support the network partitioning problem or have limitations such as poor efficiency or resource consumption Al Sadawi *et al.* [3], Wang *et al.* [4].

* Part of this work is supported by the NSF under Grant No. 185190 and 1801552 for the Florida International University REU and RET programs, respectively.

A chain-based blockchain usually aims to grow on a single chain (i.e., the longest chain) in which blocks are ordered by their creation time. When a network split occurs, two or more subnetworks can continue adding blocks to their blockchains, creating distinct blockchain copies. This scenario is possible in mobile networks, and the data from all subchains or blockchain copies could be equally important. When these subnetworks attempt to merge again, chain-based consensus algorithms usually favor selecting the blockchain copy with the longest chain and appending any future blocks on the longest chain. This approach also ignores other blocks from sub-chains or forks, which can be crucial for future use.

Traditional blockchain systems are designed in a single chain or a linear-based structure Nakamoto [1] (we will use linear-based and chain-based interchangeably). This type of structure can result in undesirable performance in terms of scalability, throughput, and confirmation time Wang *et al.* [5]. In addition, chain-based data structures can result in computation, storage and communication overheads as observed by Xu *et al.* [6]. Moreover, Li *et al.* [7] pointed out that a single chain structure can experience centralization concerns because powerful nodes have a higher chance of generating the next block. Another concern is high transaction fees because transactions are processed by powerful miners that require high resource consumption. Several graph-based structures, such as Directed Acyclic Graph or DAG-based blockchain, were introduced to address these bottlenecks. Literature improvements include developing DAG-based blockchains that can process or confirm transactions in parallel, requiring fewer communications, computations, and storage overhead [8]–[11]. Despite all the benefits of the DAG structure, a large-scale DAG-based blockchain, such as IoT scale, consumes higher computations than traditional linear-based blockchains Wang *et al.* [4].

Our contributions in this paper include developing a graph-based blockchain structure that is immune to the dynamic merge and split nature of mobile IoT systems while maintaining data consistency and trust in a trustless environment. Another contribution is designing a blockchain system that can improve scalability and efficiency over existing blockchain systems by utilizing an efficient algorithm using binary search

2324-9013/22/$31.00 ©2022 IEEE
DOI 10.1109/TrustCom56396.2022.00129

Bentley *et al.* [12] for blockchain operations.

The rest of this paper is organized as follows: Section II presents an overview and background knowledge related to this paper. Section III highlights and summarizes some of the related work. Section IV describes the preliminaries and system assumptions. Section V shows the proposed model. In Section VI, we discuss the simulation setup and experimental results. And lastly, Section VII presents the conclusion of this paper.

II. Overview

A. Blockchain

The oldest form of blockchain dates back to 1990 by Haber *et al.* [13] which describes how to cryptographically seal and time-stamp a digital document. The oldest running form of blockchain, from 1995 - present, is by the New York Times [14] using the work presented by Haber *et al.* [13]. The term blockchain became more popular with the introduction of bitcoin in 2008 by an anonymous entity known as Satoshi Nakamoto [1]. A blockchain system consists of a peer-to-peer (P2P) network where each node stores a copy of a distributed ledger (or distributed database) Hsiao *et al.* [15]. A blockchain consists of blocks that are chained together using hash values Wu *et al.* [16]. Each block can store committed digital interactions that can happen in the blockchain network Yaga *et al.* [17]. Committed digital interactions can include the amount of digital assets sent from one account to another, temperature readings, logs, or any kind of data to be stored. The blockchain grows over time by appending valid blocks to the blockchain by special nodes called miners or forgers Zheng *et al.* [18], Al Sadawi *et al.* [3]. Blockchains are also designed to achieve some of the following goals: First, eliminating the need to have a trusted third party. For example, blockchain does not require any intermediary or central authority to perform valid transactions, and nodes in blockchain systems can agree on the trustfulness of any block using consensus algorithms Dai *et al.* [19]. Additional goals include achieving user privacy by concealing users' private information while keeping records publicly available, and ensuring data integrity and immutability. There are three types of blockchains public (such as Nakamoto [1] and Ethereum [20]), private, and consortium blockchains Xu *et al.* [21].

To achieve trust in a blockchain, the system uses consensus algorithms that allow nodes to agree on any block's validity and trustfulness before appending it into the blockchain Dai *et al.* [19]. Some of the most widely used consensus algorithms are proof of work (PoW) Back *et al.* [22], Nakamoto *et al.* [1], proof of stake (PoS) Bentov *et al.* [23], delegated proof of stake (DPoS) Larimer *et al.* [24], and practical Byzantine fault tolerance (PBFT) Castro *et al.* [25]. For more details about consensus protocols, we refer to this paper Xiao *et al.* [26]. In (PoW), miners compete to solve a computationally-expensive mathematical puzzle. The more hashing power a node has, the more work it can do and the higher the chance to solve the puzzle, mine the next block, and receive the reward Wu *et al.* [16]. With blockchain systems that use PoS, the consensus algorithm requires miners or forgers to lock assets before being selected to be validators and forging any block. In general, forgers with higher locked assets (stake) are assumed to be more trustful and have higher chances to forge the next block Dorri *et al.* [27]. This paper applies PoS as a consensus algorithm since it does not require extensive computations, which is more suitable for lightweight IoT devices.

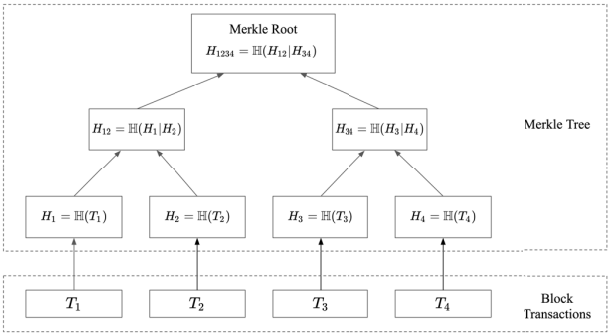

Fig. 1. Merkle Tree Structure Bao *et al.* [29].

B. Merkle Tree

A Merkle tree, proposed by Merkle *et al.* [28], is a balanced hash tree that stores hash values, see Fig. 1. Many research works utilize the Merkle tree architecture in distributed computing, such as bitcoin. Some of the Merkle tree's major applications include comparison and verification of data Bao *et al.* [29]. Every leaf node in the Merkle tree is labeled with a hash that is generated from its data content. Every non-leaf (parent) node is labeled with a hash that is generated by concatenating the hashes of its children. Hashing the tree works as a bottom-up approach starting from the leaf nodes up to the root node. In the end, a unique Merkle hash is assigned to the root node. This Merkle hash is used in blockchain systems and is stored in the block header. The final Merkle hash represents proof of the validity of all transactions within the block. One thing to note is that any modification to any transaction will result in a different Merkle root hash value Wu *et al.* [16],Wang *et al.* [4], Li *et al.* [30].

Fig. 1 depicts a simple example of the Merkle tree. The structure is divided into two parts. The first part contains the Merkle tree, and the second part is the data to be hashed, in our case, block transactions. For instance, the leaf node H_1 stores the hash value of transaction T_1, and the hash value is calculated using the hashing function $\mathbb{H}(T_1)$ using Merkle *et al.* [28], the same applies to other transactions. Moving up in the tree, every non-leaf node concatenates its children's hashes, hashes them, and stores a new hash value. In the example, the non-leaf node H_{12} concatenates H_1 and H_2 and assigns a new hash to H_{12} as follows, $H_{12} = \mathbb{H}(H_1|H_2)$. This process continues until the root node is reached, and the root node H_{1234} assigns the root hash to be $H_{1234} = \mathbb{H}(H_{12}|H_{34})$.

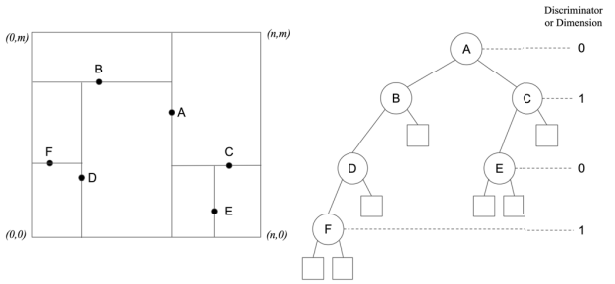

Fig. 2. A k-d tree structure with 2-dimensions Bentley *et al.* [12].

C. k-d Tree

A k-d tree, introduced in the 1970s by Bentley *et al.* [12], is a multidimensional binary search tree. The k symbol indicates the number of dimensions, formally known as discriminators. The k-d tree takes sets of inputs or points and sorts them in a k-dimensional space; see Fig. 2. The k-d tree is a useful and efficient data structure for cases like range search and nearest neighbor search Moore *et al.* [31].

Fig. 2 shows a 2-d tree which represents a set of points $\{A, ..., F\}$ in a plane. In the tree, the circles represent the points; and the squares represent $null$ leaf spaces that can accept the next input or node. At first, root node A splits the domain or points with a vertical line into two subdomains (B and C). Both subdomains in this example have an approximately equal size number of points. The splitting process continues until no splitting is required. Nodes at discriminator 0 split a set of points with a vertical line while nodes at discriminator 1 split a set of points with a horizontal line Bentley *et al.* [12].

D. Merkle k-d Tree

A Merkle k-d tree consists of a k-d tree with a Merkle tree representation of the k-d tree as described in Sections II-B and II-C. These are the main building blocks of the proposed model. The multidimensional blockchain is an immutable k-d tree, and the Merkle tree is a modifiable representation of the multidimensional blockchain, more details in Section V.

III. Related Work

Several works were proposed to address the issues of scalability, throughput, and confirmation time when using chain-based blockchain structure. Examples of these works include sharding Wang *et al.* [32], sidechain Back *et al.* [33], and cross-chain Zamyatin *et al.* [34]. In sharding, the system divides pending transactions into smaller pieces called shards in order to process them in parallel Zamani *et al.* [35]. Sidechain, an additional solution to improve traditional blockchain structure, allows digital assets to be transferred between the main chain and sidechains Wu *et al.* [16]. Cross-chain can help improve traditional blockchains by establishing communication between multiple blockchains and allowing digital assets to transfer between them. Although these approaches can enhance chain-based blockchain functionality, their backbone structures are still based on a chain-based blockchain structure Wang *et al.* [5] which is not suitable for wireless mobile networks. In addition, chain-based structures can suffer from linear growth scalability and inefficiency. The end of this section (Subsection III-A) will explain why graph-based blockchains are more suitable for mobile wireless networks than chain-based blockchains.

Shahid *et al.* [36] proposed a lightweight, scalable blockchain system for resource-constrained Internet of Things devices called "Sensor-Chain." Their proposed model allows nodes to split into multiple networks based on regions, and each network has its independent blockchain. The model also enables blockchains to merge by aggregating blocks into a single block, saving storage resources. However, the model periodically erases some historical blocks when aggregating blocks after the merge. This produces a fundamental issue because the missing historical blocks could contain critical data and negatively impact data availability and consistency. Furthermore, the model uses a chain-based blockchain structure, which is not favorable to recourse-constrained devices due to the poor overall performance metrics discussed earlier.

Due to resource limitations in IoT, many researchers proposed approaches to offload the blockchain data towards more centralized IoT resources such as edge, fog, or Roadside Units (RSU), where full nodes are located and can store the entire blockchain. A DAG-based blockchain system was proposed by Yang *et al.* [37] to enable a lightweight and secure data storage structure for resource-constrained Vehicular Social Network devices (VSNs). Another work suggested aggregating the blockchain data through base stations, roadside infrastructure, or service providers. Danzi *et al.* [38] proposed a blockchain system for lightweight IoT devices with delay and communication tradeoffs. In their work, the blockchain data is aggregated in periodic updates to reduce the communication cost of the IoT clients. Each IoT client is connected to a set of blockchain networks through wireless base stations. Memon *et al.* [39] proposed a blockchain based DualFog-IoT system with three configuration filters that can specify the type of incoming requests. These filters are Real-Time, Non-Real Time, and Delay Tolerant Blockchain applications. The DualFog architecture splits the fog layer into two parts. Fog Cloud Cluster where it communicates with the cloud, and Fog Mining Cluster which includes a group of trusted fog devices that are responsible for mining for blockchain-based applications. However, all the nodes must maintain communication with all other nodes or at least with one full node that's always connected through edge/fog devices.

Kim *et al.* [40] proposed a graph-based blockchain system, called Binary Blockchain, that can split and merge blockchains to handle the mining congestion problem. Mining congestion is a major problem in blockchains which can cause higher transaction confirmation time. The Binary Blockchain system splits chains when the load goes above a threshold and reduces the number of chains when the load goes below a threshold. However, the proposed model must maintain communication between other multiple subchains using sync blocks. These

sync blocks were introduced to ensure balance mining between multiple chains. In our proposed model, we assume that a split of forgers can work together without maintaining any type of communication with all network participants.

Geng *et al.* [41] proposed a solution for tasks accomplishment assurance in blockchain systems for IoT. The system uses a DAG-based blockchain to ensure nodes can participate in one-to-many and many-to-one tasks accomplishment without acting maliciously. The authors addressed the incapability of single-chain blockchain to support one-to-many and many-to-one dependencies. Their solution involves branching/merging of a DAG blockchain to support one-to-many and many-to-one dependencies that satisfies recognizability, compatibility, and authenticability.

Laube *et al.* [42] proposed a graph or DAG-based blockchain model where a block can have one or multiple parent/child blocks. Their work is the first to solve the split and merge problem (network partitioning problem) caused by nodes' mobility in mobile ad hoc networks (MANETs) using DAG. However, their model does not detect topology changes such as network split and only relies on flooding to disseminate data, maintain communication, and passively detect changes in topology. Not being able to actively detect topology changes can cause issues with adjusting consensus for each split Cordova *et al.* [43]. Our model can actively detect topology changes and adjust consensus accordingly, even when the network is dealing with the network partitioning problem using the work proposed by Morales *et al.* [44].

A. Summary of related works

In summary, most related works, such as [32]–[36] use a chain-based blockchain structure that stores all the data on a single chain, limiting the processing/mining of transactions/blocks due to the mining competitiveness Wang *et al.* [32]. All miners or forgers can work on a single chain, which could result in them doing the exact operations and cause wasted valuable resources such as computational power, communication, and storage. Furthermore, only working on a single chain does not allow nodes to process/add blocks to the public ledger simultaneously, affecting the system's overall throughput.Considering mobile wireless networks, they usually tend to have resource-constrained devices, making chain-based blockchain structures an unfavorable fit for them. Instead, our model uses a graph-based blockchain which has shown to have an overall better performance, as seen in Wang *et al.* [32].

Many related works, such as [37]–[39], [45], [46] assume there are fixed, powerful devices (or high connectivity) capable of continuously providing global communication to all the nodes at all times. However, due to the mobility nature of IoT and MANETs, nodes can split into smaller and independent peer-to-peer subnetworks due to the absence of edge devices. Our model is partition-tolerant and does not require high network connectivity or fixed, powerful edge devices to provide connectivity to all nodes at all times.

Current graph-based blockchains, such as DAG-based blockchains [7], [37], [42], [47], [48], have a better partition tolerance because the blockchain structure can adapt to the dynamic changes in network typologies. However, they can suffer from the following limitations. First, some rely on full connectivity to all nodes using edge devices. Second, although they can perform better than chain-based blockchains, the DAG structure does not intrinsically order blocks, but partial ordering is possible as in Karlsson *et al.* [2] and Liu *et al.* [49]. For ledgers to achieve consensus, they may need to traverse to ancestors' blocks Geng *et al.* [41]. Third, a large-scale DAG-based blockchain consumes higher computational resources than traditional linear-based blockchains Wang *et al.* [4]. Our model can always order blocks which can help facilitate the split and merge of blockchains by efficiently scanning and adding blocks between multiple blockchains.

IV. PRELIMINARIES

This section presents an overview of the blockchain model, notations, assumptions, and proposed model.

The blockchain model uses Proof-of-Stake (PoS) for the consensus mechanism. Proof-of-Work (PoW) is considered a computationally expensive consensus algorithm; therefore, PoS seems more logical for mobile, wireless, and lightweight devices because it does not require a lot of computation to forge the next block. Instead of competing to forge the next block, a node (forger) is selected to forge the next block at fixed time interval fT (check Table I for notations). The selection is based on uniform random mining, and every node at the initialization has the same chance to mine the next block. The forger gathers all transactions, prepares an $mkdBlock$, adds and broadcasts the block to all peers in the network, in which every peer will verify the new block and add the block to the blockchain $\mathbb{T}$.

A. System Model and Assumptions

The proposed model consists of a set of IoT devices and a plane that is divided into smaller regions called cells. Each cell can contain a set of IoT devices and a local blockchain. A blockchain split happens when a group of nodes tries to split into two or more groups, each with its dedicated cell. A merge can happen when certain conditions occur: 1) when two or more local networks meet in a single cell. 2) when a local network gets access to a full node that stores the entire blockchain and is always connected to the Internet using, for example, RSU or edge devices Cordova *et al.* [43]. To ensure the authenticity and integrity of transactions in a graph-based blockchain, the block creator signs all transactions within the block as in Karlsson *et al.* [2]. Transactions contain sensor readings such as temperature readings. Another assumption is a single or multiple nodes can move from one cell to another. Our model can actively detect topology changes and adjust consensus accordingly even when the network is dealing with the network partitioning problem using Morales *et al.* [44]. Each local blockchain network can work independently without relying on other blockchain networks. Nodes are assumed

TABLE I
NOTATIONS

Symbol	Meaning
$\mathbb{T}$	Merkle k-d blockchain (MKDBC)
$block$	An arbitrary block
$mkdBlock$	A Merkle k-d block in a $\mathbb{T}$
$hash$	The hash of $block$ or $mkdBlock$
$mhash$	The Merkle hash of $mkdBlock$
tx	Transaction abbreviation
cT	Current time
gT	Genesis time
fT	Forge time interval
C	A cell or region
$\mathbb{C}$	Set of all cells
sC	spatial constraint
s	A sensor
S	A set of sensors
$\mathbb{S}$	Set of all sensors
G	local network
G_i^t	A local network of nodes in cell i at time t
$\mathbb{G}$	Set of all local networks
n	Total number of sensors
V_i^t	Set of vertices of local network G_i^t
$\mathbb{T}_i^t$	A local Merkle k-d blockchain in network G_i^t
$\mathbb{T}^{gT}$	MKD blockchain at genesis time gT
$cDim$	Current dimension
$tDim$	Total number of dimensions of $\mathbb{T}$, ranging from 0 to k

to be mobile within a cell using Random Way-point Mobility found in Hyyti *et al.* [50] and Reference Point Group Mobility (RPGM) for group trajectory based on Hong *et al.* [51]. Each node is capable of performing simple data aggregation tasks such as finding max, min, mean, etc. Pumpichet *et al.* [52]. The proposed model does not require any additional powerful devices since nodes participating in the blockchain can perform all the necessary operations. In addition, the model utilizes a permissioned version of blockchain where a centralized authority controls who participates in the blockchain and assigns a public and private key for each node. Each node has the same genesis block. We also assume that nodes reveal their identities to each other using a privacy-preserving method such as Li *et al.* [53]. The nodes achieve consensus using proof of stake (PoS) consensus algorithm as in Bentov *et al.* [23]. The consensus algorithm is set to have finality, which means the consensus protocol does not allow the presence of equally valid blocks or subchains. This is achievable using many approaches, such as applying three consensus phases before committing any request. The three phases are: pre-prepare, prepare, and commit as in Xing *et al.* [54], Castro *et al.* [25]. Another approach is the NEAR protocol [55] which can achieve finality in one round of communication. And lastly, the work by Ethereum [56] to implement single-slot finality for Ethereum in around 16 seconds.

V. THE MULTIDIMENSIONAL OR MERKLE k-D BLOCKCHAIN FRAMEWORK

The Merkle k-d blockchain model (MKDBC) consists of two major components, as shown in Fig. 3. The first one is a multidimensional blockchain. Unlike traditional blockchain models where blocks are structured as a linear-chain, the MKDBC structures and sorts blocks in a way similar to a k-d tree Bentley *et al.* [12], see Section II-C. The (genesis block, previous block) in MKDBC have the same analogy as the (root, parent) in k-d trees, respectively. The block components of MKDBC are similar to any other traditional block except in the block header; we have an extra field that stores the block dimensions as a list of k values where k is the number of dimensions of the blockchain or $[Dim_0, Dim_1, ..., Dim_k]$.

The second component of MKDBC is a Merkle tree representation of the entire blockchain using the method designed by Merkle *et al.* [28], see Section II-B. The primary goals of using the Merkle tree are to provide an extra layer of security and as a tool for fast comparison and verification of blocks; more details in Section VI. Each time a block is added to the multidimensional blockchain, the Merkle tree gets updated. Generating and updating the Merkle hash tree is an efficient process since it only needs a few modifications to the tree rather than searching and updating the entire tree Merkle *et al.* [28]. The Merkle tree is stored separately from the blockchain since the Merkle tree is not immutable and requires frequent updates each time a block is added to the blockchain.

Because nodes are highly mobile in mobile wireless networks, our model utilizes spatiotemporal data, such as coordinates and time, as discriminators or dimensions for the multidimensional blockchain. Although it's possible to adjust the blockchain dimensions based on the application, we decided to use coordinates and time dimensions for simplicity and the randomness of nodes' movements. The randomness generated from node movements turned out to be helpful in balancing the multidimensional tree; more explanations about this are in Section. VI. The process of blockchain split and merge is shown in Fig. 4. The rest of this section will explain the algorithms of our model.

Algorithm 1, lines 1 - 5, start with initializing an empty Merkle k-d blockchain or $\mathbb{T}$. Later, the

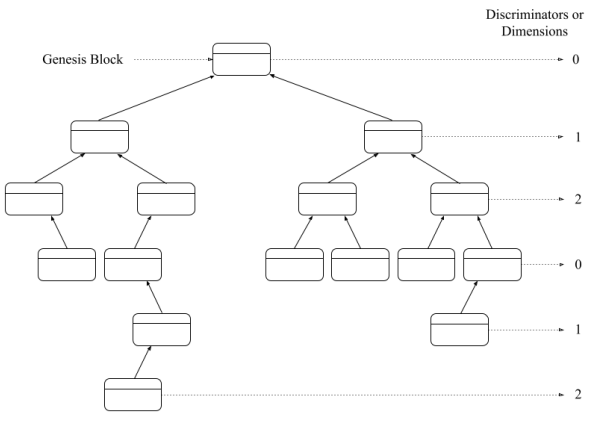

Fig. 3. The multidimensional blockchain structure with 3 dimensions or $k = 3$, the dimensions are $0, 1, 2$.

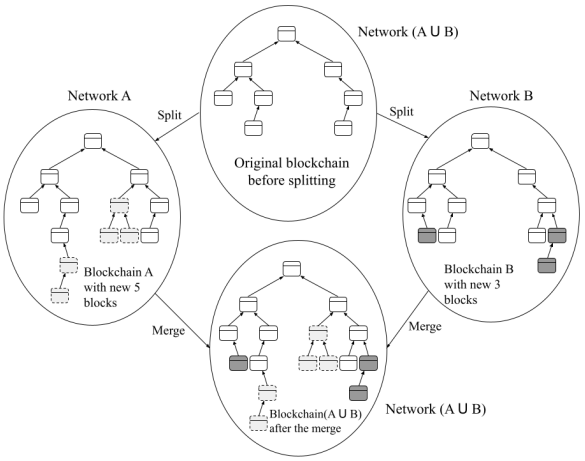

Fig. 4. The process of split and merge in multidimensional blockchains.

Algorithm 1 MKDBC Management

Input: Current time: cT. Genesis time: gT. Forge time interval: fT. Set of all local networks: $\mathbb{G}$.
Output: Updated MKDBC Tree: $\mathbb{T}$

1: $\mathbb{T} \leftarrow \varnothing$
2: $insertMkdBlock(\mathbb{T}, mkdGenesisBlock)$ See Algorithm 3
3: $updateMerkelHash(\mathbb{T}, mkdGenesisBlock)$
4: **for** each $s \in \mathbb{S}$ **do**
5: $download(\mathbb{T}^{gT})$
6: **end for**
7: **if** $(cT - gT) \mod fT == 0$ **then**
8: **for** each $G_i^t \in \mathbb{G}^t$ **do**
9: $Forger \leftarrow selectForger(V_i^t)$
10: $newMkdBlock \leftarrow creatMkdBlock(Forger)$
11: $insertMkdBlock(\mathbb{T}_i^t, newMkdBlock)$, See Algorithm 3
12: **end for**
13: **end if**

blockchain gets the first mkdBlock as a genesis block using $insertMkdBlock$ function, and the Merkle hash is updated using $updateMerkleHash$. Once the genesis block is added to the blockchain and the Merkle hash is updated, every node will download a copy of the blockchain at the genesis time or $\mathbb{T}^{gT}$. As mentioned in Subsection IV-A, for any node to join the network, the node is required to have a copy of the blockchain at the genesis time or $\mathbb{T}^{gT}$. This is possible because we are using a permissioned type of blockchain where there is a centralized entity controlling who joins the network. The reason why we require a node to have at least a copy of $\mathbb{T}^{gT}$ is to apply some balance to the blockchain tree as it grows in size, which will be explained further in Section VI-A. Finally, lines 7 - 11 show whenever it is time to forge a new block, a forger will be selected in each local network G_i^t to forge the next mkdBlock.

Algorithm 2 MKDBC Management During Mobility

Input: Two MKD Blockchain Trees: $\mathbb{T}$. Set of local networks: G. Set of cells: C. Set of sensors: S
Output: Updated MKD Blockchain Tree: $\mathbb{T}$

1: **if** $S.C_{cur} \neq C_{new\_cell}$ **then**
2: $\mathbb{T}_m \leftarrow blockchainToMerge(\mathbb{T}_{cur}, \mathbb{T}_{new\_cell})$
3: $\mathbb{T}_k \leftarrow blockchainToKeep(\mathbb{T}_{cur}, \mathbb{T}_{new\_cell})$
4: $Aggregator \leftarrow selectAggregator(V_m^t, V_k^t)$
5: **for** each $mkdBlock_m \in \mathbb{T}_m^t$ **do**
6: **if** $mkdBlock_m \notin \mathbb{T}_k^t$ **then**
7: $newMkdBlock_m \leftarrow creatMkdBlock(Aggregator)$
8: $insertMkdBlock(\mathbb{T}_k^t, newMkdBlock_m)$, See Algorithm 3
9: **end if**
10: **end for**
11: $delete(\mathbb{T}_m^t)$ from either $S.C_{cur}$ or C_{new_cell}
12: $S.C_{cur} \leftarrow C_{new\_cell}$
13: $G_{new\_new}^t \leftarrow G_{new\_new}^t \cup G_{curr}^t$
14: $V_m^t.download(\mathbb{T}_k^t)$ from peers V_k^t
15: **end if**

Algorithm 2 handles nodes' mobility management or when nodes are moving from one region to another. Lines 1 - 3 check if a set of nodes $S.C_{cur}$ is joining a another cell C_{new_cell}. First, the algorithm decides which blockchain to keep $\mathbb{T}_k$ and which blockchain to merge $\mathbb{T}_m$. There are many factors to choose from, but for simplicity, we chose to merge blockchains based on their sizes or to merge the smaller blockchain with the larger one. Secondly, after deciding which blockchain to keep and which one to merge, lines 4 - 8 select an aggregator node to handle the blockchain merge. Each block in $\mathbb{T}_m$ will be scanned and merged, if needed, into $\mathbb{T}_k$. The merge also utilizes the same method of adding blocks as in Algorithm 3. Lines 11 - 14 delete the blockchain from either the current or new cell, depending on which blockchain to merge and keep, update their cell/network information, and download the blockchain $\mathbb{T}_k$ after the merge is complete.

Algorithm 3 describes a recursive function $insertBlock$ which inserts or forges $block$ into the Merkle k-d blockchain $\mathbb{T}$ as an $mkdBlock$. The insertion method is a modified version from the k-d tree insertion found in Section II-C or Bentley *et al.* [12]. Algorithm 3 starts by taking multiple inputs which include the MKDBC tree: $\mathbb{T}$, the block to be forged: $block$, a parent block to compare with: $mkdBlock$, current dimension: $cDim$, and the total number of dimensions of $\mathbb{T}$: $tDim$. At the first iteration, lines 1 - 3 check if the first parent $mkdBlock$ or $T.root$ is $null$, however, the model assumes all nodes have the same genesis mkdBlock and hence the algorithm will skip lines 1 - 3. Next, at line 4, the algorithm checks whether $block$ is a duplicate at the current dimension in $\mathbb{T}$ or not. Later, at line 6 - 9, the algorithm continues to recursively scan the tree cycling between each dimension until it hits a $null$ or a space that accepts the next block. The algorithm hits a $null$ leaf at line 1 where $block$ will be forged and placed in $\mathbb{T}$ at the proper

Algorithm 3 insertMkdBlock

Input: MKDBC Tree: $\mathbb{T}$. The block to be forged: $block$. A parent block to compare with: $mkdBlock$. Current dimension: $cDim$. Total number of dimension of $\mathbb{T}$: $tDim$
Output: MKDBC Tree with the new forged block: $\mathbb{T}$

1: **if** $mkdBlock == null$ **then**
2: $mkdBlock \leftarrow forgeBlock(block)$
3: $updateMerkleHash(\mathbb{T}, mkdBlock)$, See algorithm 4
4: **else if** $block == mkdBlock.data$ **then**
5: **return** 'duplicate block'
6: **else if** $block[cDim] < mkdBlock[cDim]$ **then**
7: $mkdBlock.left \leftarrow insertMkdBlock(\mathbb{T}, block, mkdBlock.left, (cDim + 1) \bmod tDim,\ tDim)$
8: **else**
9: $mkdBlock.right \leftarrow insertMkdBlock(\mathbb{T}, block, mkdBlock.right, (cDim+ 1) \bmod tDim,\ tDim)$
10: **end if**

location.

Algorithm 4 presents a recursive function to update the Merkle hash of the entire MKDBC tree. The update function is called immediately after forging any new block. Calculating or updating the Merkle hash is an efficient process and does not require scanning or updating the entire tree but rather one branch of the tree Merkle *et al.* [28]. The update function takes the MKDBC tree $\mathbb{T}$ and an arbitrary $block$ or $mkdBlock$ in $\mathbb{T}$. At line 1, the algorithm starts with finding the $parent$ block of $block$ in $\mathbb{T}$. The method used to find the $parent$ block is similar to the k-d search as in Section II-C. Finding the $parent$ block also gives access to the $parent$'s child blocks. Lines 2 - 11 check if $block$ is the root of $\mathbb{T}$; if the condition is met, the function terminates since there is no need to update the Merkle hash of $\mathbb{T}$. Next, if $block$ is not the root of $\mathbb{T}$, $block$ will be compared to check if it is the right or left child of $parent$ and update $T.parent.hash$ accordingly using the method in Section II-B. In summary, the recursive function $updateMerkleHash(T, block)$ starts from the most recently forged block, and applies all the necessary Merkle hash updates to the root or genesis block.

VI. Evaluation

A. Simulation Setup

For the simulation, we used 80 nodes on a grid with size $m \times n$. All nodes get a copy of the initial blockchain, which only includes the genesis block. Blocks are similar to any other traditional blockchain except for an extra field that stores the block dimensions; the dimensions used are $[x\_coordinate, y\_coordinate, time]$. The order of dimensions is critical, as we will see later. The main goal of ordering is to allow the multidimensional blockchain to grow as balanced as possible on both sides of the genesis block. To achieve some balance in the blockchain tree, we set the

Algorithm 4 updateMerkleHash

Input: MKDBC Tree: $\mathbb{T}$, An arbitrary block: $block$
Output: Updated MKDBC Tree: $\mathbb{T}$

1: $parent \leftarrow getParentBlock(T, block)$, See Section II-C
2: **if** $block == \mathbb{T}.root$ **then**
3: **return**
4: **else if** $block == \mathbb{T}.parent.left$ **then**
5: $\mathbb{T}.parent.mhash \leftarrow$
6: $hash(block.mhash|\mathbb{T}.parent.right.mhash)$, See Section II-B
7: **return** $updateMerkleHash(\mathbb{T}, parent)$
8: **else**
9: $\mathbb{T}.parent.mhash \leftarrow$
10: $hash(\mathbb{T}.parent.left.mhash|block.mhash)$
11: **return** $updateMerkleHash(\mathbb{T}, parent)$
12: **end if**

genesis block dimensions, in this case, the coordinates, as the mid-point of the grid or 50. Since nodes are highly mobile and travel randomly within the grid, future blocks can be added on both sides of the genesis block. In addition, to allow additional balance to the blockchain tree, we set $time$ as the last in the dimension list $[x\_coordinate, y\_coordinate, time]$. This is crucial because time is an incremental value, and if we set time to be the first dimension, the blockchain will only grow on one side of the blockchain tree, in this case, the right side. If $time$ is the first dimension, the genesis block will have a $time$ dimension set to 0, and any future block will have $time$ dimension > 0 and hence will be added to the right side of the blockchain tree.

For node mobility, we implemented two mobility models, Reference Point Group Mobility (RPGM) based on this work Hong *et al.* [51] and Random Waypoint Mobility Bettstetter *et al.* [57]. In RPGM, the nodes within a group are uniformly distributed inside a circle, and the circle center represents the group center Sichitiu *et al.* [58]. In addition, the group center travels on a random trajectory, and all nodes move at a random velocity ranging from 0 to 1. The code to generate these types of movements is publicly available on GitHub [59].

B. Experimental Results

Fig. 5 shows a summary of 835,000 blockchain snapshots captured throughout the simulation. The goal is to find the maximum chain length or height in the multidimensional blockchain and compare it with: 1) the traditional chain-based blockchain height such as Nakamoto [1] and 2) a balanced k-d tree height Bentley *et al.* [12] (best case scenario). It's essential to consider the blockchain height because having longer branches results in more comparisons (i.e., more resource consumption) to do any operation, such as appending/merging/scanning blocks, in the multidimensional blockchain. First, we grouped the blockchain snapshots based on their blockchain sizes in an increment of 50 where group one includes all the blockchains with sizes $\geqslant 0$ and $\leqslant 49$ and so on. Then for each group, we calculated the maximum chain

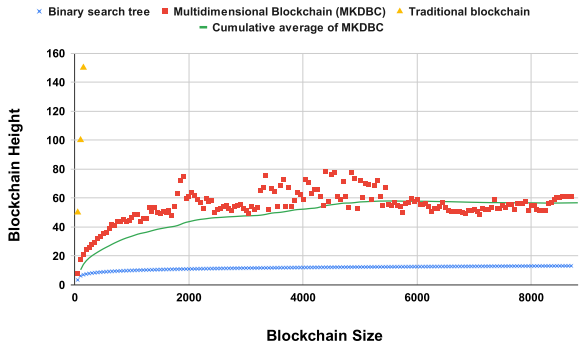

Fig. 5. The longest chain height or length, averaged per blockchain size for the multidimensional blockchain (MKDBC), compared with a traditional chain-based blockchain [1], and a balanced binary tree [12]. Lastly, the cumulative average of MKDBC longest height is represented by the line

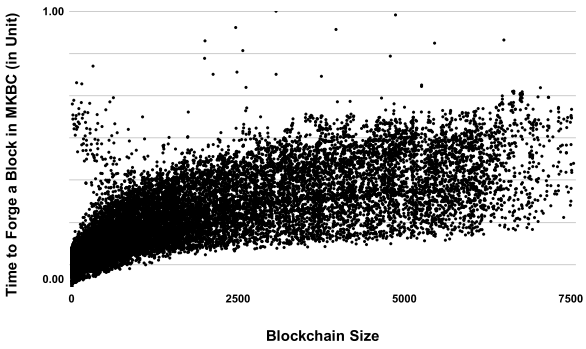

Fig. 6. Time to forge a block in the multidimensional blockchain (MKDBC), normalized to a unit of time where 1 is the maximum recorded time.

length or height per snapshot and found the total average of the maximum chain length of the group. Based on Fig.5, the average maximum chain height, of all MKDBC blockchains with a size around 8000 blocks, is roughly 55. And the maximum chain height, of a traditional blockchain with 8000 blocks, is 8000. Comparing the maximum height of MKDBC with the traditional chain-based blockchain, we can see that MKDBC can reduce the maximum chain length by more than 99.99%. This means, on average, it only scans less 0.01% of the total blocks to find the place to forge or merge the next block in MKDBC. This reduction allows the multidimensional blockchain to perform efficiently as the blockchain grows; the next Fig. 6 will demonstrate this. Some could argue that a smaller sub-chain length (leaf block to genesis block) can allow malicious nodes to redo the work of that particular sub-chain. This is one of the reasons we use a Merkle tree representation of the multidimensional blockchain, which is to verify whether a sub-chain is valid relative to the entire blockchain or has been compromised. Any modification to any sub-chain will result in a different Merkle hash for the entire multidimensional blockchain. An important observation is that traditional blockchain systems' height or chain length grows linearly, and the chart can only cover a height up to 150 for a blockchain with 150 blocks. However, the MKDBC height stays relatively similar to a logarithmic growth, showing signs of efficiency and scalability as the model grows.

The following evaluation chart or Fig. 6 shows the time the MKDBC takes to forge or merge a block in a unit of time (normalized), where 1 is the maximum recorded time. The dots represent the forging time of more than 20,000 random blocks from random blockchains of different sizes. Unlike traditional blockchain, where blocks are appended to the end of the chain with almost no time, our model requires searching the multidimensional blockchain for the right location to append the next block. Luckily, the MKDBC utilizes a binary search operation to find the right place to forge the next block. The details on how to search and forge the next block are explained in the recursive Algorithm 3. As shown in Fig. 6, the time to search and forge blocks shows an efficient and scalable growth. Finally, we will point out observations using the same graph to demonstrate another practical use of multidimensional blockchain for mobile IoT with network partitioning problems. Fig. 6 shows a cluster of blocks, between blockchains with sizes ranging between 1 - 1000, with higher than normal forging time. The reason is that those blocks were forged in blockchains that stayed relatively stationary in particular areas (we will call them stationary blockchains). Since we're using coordinates as the first and the second dimensions, the blockchain continued to grow largely on one side of the multidimensional blockchain, resulting in longer than normal sub-chains or branches (More details in Section VI-A). The higher the branches, the more time it takes to forge or merge blocks since it involves more traversing and comparisons. Interestingly, the forging time declines as those stationary blockchains start to grow, move, and merge with other blockchains. The transition from being stationary blockchains to becoming more active or mobile can help in adding additional balancing to the blockchain tree and produce blockchains with shorter branches or heights, resulting in improving the forging time.

VII. Conclusion

This paper presents a multidimensional graph-based blockchain model that eliminates the need for having fixed and powerful peripherals for mobile IoT. The model can allow a blockchain system to function even when dealing with the network partitioning problem while maintaining the blockchain's security and privacy. Experimental results show the multidimensional blockchain can achieve: efficiency by having to scan only a few blocks (fewer comparisons) for any blockchain operations, and scalability by achieving performance similar to logarithmic growth. In future work, we plan to study the effect of dimensionalities on the performance of the multidimensional blockchain. We will investigate the impact by experimenting with different numbers and kinds of dimensions.

REFERENCES

[1] S. Nakamoto *et al.*, "Bitcoin: A peer-to-peer electronic cash system.(2008)," 2008.

[2] K. Karlsson, W. Jiang, S. Wicker, D. Adams, E. Ma, R. van Renesse, and H. Weatherspoon, "Vegvisir: A partition-tolerant blockchain for the internet-of-things," in *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2018, pp. 1150–1158.

[3] A. Al Sadawi, M. S. Hassan, and M. Ndiaye, "A survey on the integration of blockchain with iot to enhance performance and eliminate challenges," *IEEE Access*, vol. 9, pp. 54 478–54 497, 2021.

[4] W. Wang, D. T. Hoang, P. Hu, Z. Xiong, D. Niyato, P. Wang, Y. Wen, and D. I. Kim, "A survey on consensus mechanisms and mining strategy management in blockchain networks," *Ieee Access*, vol. 7, pp. 22 328–22 370, 2019.

[5] Q. Wang, J. Yu, S. Chen, and Y. Xiang, "Sok: Diving into dag-based blockchain systems," *arXiv preprint arXiv:2012.06128*, 2020.

[6] M. Xu, C. Liu, Y. Zou, F. Zhao, J. Yu, and X. Cheng, "wchain: a fast fault-tolerant blockchain protocol for multihop wireless networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 10, pp. 6915–6926, 2021.

[7] Y. Li, B. Cao, M. Peng, L. Zhang, L. Zhang, D. Feng, and J. Yu, "Direct acyclic graph-based ledger for internet of things: Performance and security analysis," *IEEE/ACM Transactions on Networking*, vol. 28, no. 4, pp. 1643–1656, 2020.

[8] Y. Sompolinsky, Y. Lewenberg, and A. Zohar, "Spectre: A fast and scalable cryptocurrency protocol," *Cryptology ePrint Archive*, 2016.

[9] H. Pervez, M. Muneeb, M. U. Irfan, and I. U. Haq, "A comparative analysis of dag-based blockchain architectures," in *2018 12th International conference on open source systems and technologies (ICOSST)*. IEEE, 2018, pp. 27–34.

[10] G. Birmpas, E. Koutsoupias, P. Lazos, and F. J. Marmolejo-Cossío, "Fairness and efficiency in dag-based cryptocurrencies," in *International Conference on Financial Cryptography and Data Security*. Springer, 2020, pp. 79–96.

[11] Q. Zhou, H. Huang, Z. Zheng, and J. Bian, "Solutions to scalability of blockchain: A survey," *Ieee Access*, vol. 8, pp. 16 440–16 455, 2020.

[12] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[13] S. Haber and W. S. Stornetta, "How to time-stamp a digital document," in *Conference on the Theory and Application of Cryptography*. Springer, 1990, pp. 437–455.

[14] H. Treiblmaier and T. Clohessy, *Blockchain and Distributed Ledger Technology Use Cases*. Springer, 2020.

[15] S.-J. Hsiao and W.-T. Sung, "Employing blockchain technology to strengthen security of wireless sensor networks," *IEEE Access*, vol. 9, pp. 72 326–72 341, 2021.

[16] M. Wu, K. Wang, X. Cai, S. Guo, M. Guo, and C. Rong, "A comprehensive survey of blockchain: From theory to iot applications and beyond," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8114–8154, 2019.

[17] D. Yaga, P. Mell, N. Roby, and K. Scarfone, "Blockchain technology overview," *arXiv preprint arXiv:1906.11078*, 2019.

[18] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *International Journal of Web and Grid Services*, vol. 14, no. 4, pp. 352–375, 2018.

[19] H.-N. Dai, Z. Zheng, and Y. Zhang, "Blockchain for internet of things: A survey," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8076–8094, 2019.

[20] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[21] X. Xu, I. Weber, M. Staples, L. Zhu, J. Bosch, L. Bass, C. Pautasso, and P. Rimba, "A taxonomy of blockchain-based systems for architecture design," in *2017 IEEE International Conference on Software Architecture (ICSA)*. IEEE, 2017, pp. 243–252.

[22] A. Back *et al.*, "Hashcash-a denial of service counter-measure," 2002.

[23] I. Bentov, R. Pass, and E. Shi, "Snow white: Provably secure proofs of stake." *IACR Cryptol. ePrint Arch.*, vol. 2016, p. 919, 2016.

[24] D. Larimer, "Delegated proof-of-stake (dpos)," *Bitshare whitepaper*, 2014.

[25] M. Castro, B. Liskov *et al.*, "Practical byzantine fault tolerance," in *OSDI*, vol. 99, no. 1999, 1999, pp. 173–186.

[26] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou, "A survey of distributed consensus protocols for blockchain networks," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1432–1465, 2020.

[27] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram, "Lsb: A lightweight scalable blockchain for iot security and anonymity," *Journal of Parallel and Distributed Computing*, vol. 134, pp. 180–197, 2019.

[28] R. C. Merkle, "A certified digital signature," in *Conference on the Theory and Application of Cryptology*. Springer, 1989, pp. 218–238.

[29] Z. Bao, W. Shi, D. He, and K.-K. R. Chood, "Iotchain: A three-tier blockchain-based iot security architecture," *arXiv preprint arXiv:1806.02008*, 2018.

[30] F. Li, K. Yi, M. Hadjieleftheriou, and G. Kollios, "Proof-infused streams: Enabling authentication of sliding window queries on streams," in *Proceedings of the 33rd international conference on Very large data bases*, 2007, pp. 147–158.

[31] A. W. Moore, "Efficient memory-based learning for robot control," Ph.D. dissertation, 1990.

[32] G. Wang, Z. J. Shi, M. Nixon, and S. Han, "Sok: Sharding on blockchain," in *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, 2019, pp. 41–61.

[33] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Timón, and P. Wuille, "Enabling blockchain innovations with pegged sidechains," *URL: http://www.opensciencereview. com/papers/123/enablingblockchain-innovations-with-pegged-sidechains*, vol. 72, 2014.

[34] A. Zamyatin, M. Al-Bassam, D. Zindros, E. Kokoris-Kogias, P. Moreno-Sanchez, A. Kiayias, and W. J. Knottenbelt, "Sok: communication across distributed ledgers." 2019.

[35] M. Zamani, M. Movahedi, and M. Raykova, "Rapidchain: Scaling blockchain via full sharding," in *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, 2018, pp. 931–948.

[36] A. R. Shahid, N. Pissinou, C. Staier, and R. Kwan, "Sensor-chain: a lightweight scalable blockchain framework for internet of things," in *2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, 2019, pp. 1154–1161.

[37] W. Yang, X. Dai, J. Xiao, and H. Jin, "Ldv: A lightweight dag-based blockchain for vehicular social networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 6, pp. 5749–5759, 2020.

[38] P. Danzi, A. E. Kalør, Č. Stefanović, and P. Popovski, "Delay and communication tradeoffs for blockchain systems with lightweight iot clients," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2354–2365, 2019.

[39] R. A. Memon, J. P. Li, M. I. Nazeer, A. N. Khan, and J. Ahmed, "Dualfog-iot: Additional fog layer for solving blockchain integration problem in internet of things," *IEEE Access*, vol. 7, pp. 169 073–169 093, 2019.

[40] Y. Kim and J. Jo, "Binary blockchain: Solving the mining congestion problem by dynamically adjusting the mining capacity," in *International Conference on Applied Computing and Information Technology*. Springer, 2017, pp. 29–49.

[41] T. Geng, L. Njilla, and C.-T. Huang, "Smart markers in smart contracts: Enabling multiway branching and merging in blockchain for decentralized runtime verification," in *2021 IEEE Conference on Dependable and Secure Computing (DSC)*. IEEE, 2021, pp. 1–8.

[42] A. Laube, S. Martin, and K. Al Agha, "A solution to the split & merge problem for blockchain-based applications in ad hoc networks," in *2019 8th International Conference on Performance Evaluation and Modeling in Wired and Wireless Networks (PEMWN)*. IEEE, 2019, pp. 1–6.

[43] D. Cordova, A. Laube, G. Pujolle *et al.*, "Blockgraph: A blockchain for mobile ad hoc networks," in *2020 4th Cyber Security in Networking Conference (CSNet)*. IEEE, 2020, pp. 1–8.

[44] D. C. Morales, P. B. Velloso, A. Laube, G. Pujolle *et al.*, "C4m: A partition-robust consensus algorithm for blockgraph in mesh network," in *2021 5th Cyber Security in Networking Conference (CSNet)*. IEEE, 2021, pp. 82–89.

[45] M. Xu, F. Zhao, Y. Zou, C. Liu, X. Cheng, and F. Dressler, "Blown: a blockchain protocol for single-hop wireless networks under adversarial sinr," *IEEE Transactions on Mobile Computing*, 2022.

[46] J. Yuan and L. Njilla, "Lightweight and reliable decentralized reward system using blockchain," in *IEEE INFOCOM 2021-IEEE Confer-*

ence on Computer Communications Workshops (INFOCOM WKSHPS). IEEE, 2021, pp. 1–6.
[47] M. Cao, L. Zhang, and B. Cao, "Toward on-device federated learning: a direct acyclic graph-based blockchain approach," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.
[48] F. Guo, F. R. Yu, H. Zhang, H. Ji, M. Liu, and V. C. Leung, "Adaptive resource allocation in future wireless networks with blockchain and mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 19, no. 3, pp. 1689–1703, 2019.
[49] Y. Liu, K. Wang, K. Qian, M. Du, and S. Guo, "Tornado: Enabling blockchain in heterogeneous internet of things through a space-structured approach," *IEEE Internet of Things Journal*, vol. 7, no. 2, pp. 1273–1286, 2019.
[50] E. Hyytiä and J. Virtamo, "Random waypoint mobility model in cellular networks," *Wireless Networks*, vol. 13, no. 2, pp. 177–188, 2007.
[51] X. Hong, M. Gerla, G. Pei, and C.-C. Chiang, "A group mobility model for ad hoc wireless networks," in *Proceedings of the 2nd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, 1999, pp. 53–60.
[52] S. Pumpichet, N. Pissinou, X. Jin, and D. Pan, "Belief-based cleaning in trajectory sensor streams," in *2012 IEEE International Conference on Communications (ICC)*. IEEE, 2012, pp. 208–212.
[53] L. Li, J. Liu, L. Cheng, S. Qiu, W. Wang, X. Zhang, and Z. Zhang, "Creditcoin: A privacy-preserving blockchain-based incentive announcement network for communications of smart vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 7, pp. 2204–2220, 2018.
[54] J. Xing, D. Fischer, N. Labh, R. Piersma, B. C. Lee, Y. A. Xia, T. Sahai, and V. Tarokh, "Talaria: A framework for simulation of permissioned blockchains for logistics and beyond," *arXiv preprint arXiv:2103.02260*, 2021.
[55] A. Skidanov, "Doomslug vs pbft, tendermint, and hotstuff," Feb 2020. [Online]. Available: https://near.org/blog/doomslug-comparison/
[56] Ethereum.org, "Paths toward single-slot finality." [Online]. Available: https://notes.ethereum.org/@vbuterin/single_slot_finality
[57] C. Bettstetter, H. Hartenstein, and X. Pérez-Costa, "Stochastic properties of the random waypoint mobility model," *Wireless networks*, vol. 10, no. 5, pp. 555–567, 2004.
[58] M. L. Sichitiu, "Mobility models for ad hoc networks," in *Guide to Wireless Ad Hoc Networks*. Springer, 2009, pp. 237–254.
[59] A. Panisson. [Online]. Available: https://github.com/panisson/pymobility