Shadows: Blockchain Virtualization for Interoperable Computations in IIoT Environments

Riya Tapwal, Pallav Kumar Deb, *Graduate Student Member, IEEE*, Sudip Misra, *Fellow, IEEE*, Surjya Kanta Pal

***Abstract*—In this work, we propose *Shadows*, a virtual blockchain (VC) for achieving parallel consensus and efficient management of data in industries by utilizing BC. Typically, industrial processes involve heterogeneous activities which require real-time consensus, managed execution, isolation, data sharing, accelerated computation, and efficient utilization of various computational resources such as CPU, RAM, and storage. Achieving these in real-time using a single conventional blockchain (BC) leads to the exertion of computational power. To achieve resource-efficient real-time consensus, we virtualize the nodes of the BC network and create different BC for various activities. Further, to virtualize BC and provide better access to data, we propose smart contracts liable for providing a unified view of a single BC, dynamically creating BCs, allocating resources to these, and making communication between the same. Through lab-scale experiments, we demonstrate that *Shadows* is capable of utilizing the resources efficiently and achieving real-time consensus. In particular, *Shadows* uses 18% CPU and 92% memory while reducing consensus time by 56%, compared to a single conventional BC. *Shadows* also accesses the data efficiently by utilizing smart contracts and dynamically balances the load by migrating the virtual nodes. Further, *Shadows* reduces the number of migrations to make the balance system by 67%.**

***Index Terms*—Blockchain, Industrial Internet of Things, Virtualization, Resource allocation, Smart contracts, Osmotic Computing, Parallel consensus.**

I. Introduction

The heterogeneous functionalities and activities of Industry 4.0 in different application scenarios lead to data management and security challenges. Toward achieving this, the blockchain (BC) system is an ideal platform that offers *data security, transparency, and immutability* [1]. However, the divergent activities in industries require *managed execution, isolation, sharing of data, real-time computation, and efficient storage* which a single conventional BC (CBC) is unable to provide. Utilizing CBC leads to dormant consensus and adversity in data storage. Further, utilizing separate networks of nodes for creating BC for different activities leads to the wastage of computational resources such as CPU, RAM, and storage and decelerates the computation of consensus. In such scenarios, the virtualization of nodes in the BC network for facilitating parallel consensus and managed execution of data from different activities is essential. Further, the need for utilization of resources as well as isolation is essential for industries with heterogeneous activities. Apart from this, utilizing CBC leads to delays in accessing the data and difficulty in managing it, which necessitates the need for data distribution among multiple BC and accessing the same efficiently. In this work, we propose *Shadows*, a virtual blockchain (VC) for guided interoperability among various activities by storing the data over various blockchains (BCs) in a parallel manner and addressing the challenges specified earlier. The essence of this work is the competence of virtualization to utilize the resources more efficiently and facilitate the computation of consensus in a parallel manner, in addition to security and efficient storage. As shown in Fig. 1, we virtualize the nodes in the BC network by deploying various virtual machines on different nodes, which are liable for achieving consensus in a parallel manner and generating separate BC for heterogeneous activities. To empower *Shadows*, we create three smart contracts: 1). Antumbra (Authenticative smart contract) 2). Penumbra (Selective smart contract) 3). Umbra (Communicative smart contract). Antumbra is liable for dynamically generating BC, allocating resources among them, and providing an outer view of VC to access it, Penumbra is responsible for selecting the appropriate BC for storing transactions and partially stores the data of industries in each BC, and Umbra facilitates data sharing, provides a unified view of single BC to the end-users and completely hides the computation and storage details from the end-users. Further, we also utilize osmotic computing [2] to balance the load among various virtual nodes handling different BCs. In summary, *Shadows* does not utilize a single BC for managing data, which reduces the dormant consensus and resource wastage. Further, as the virtual BC is just a unified view of all the heterogeneous BC, the data is reflected

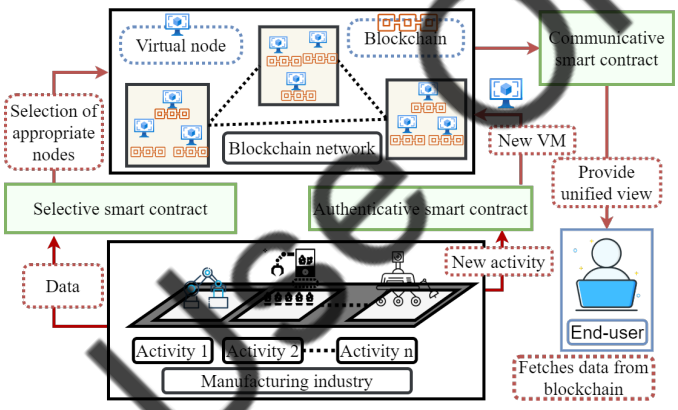


Figure 1: An overview of *Shadows'* architecture

R. Tapwal, P. K. Deb, and S. Misra are with the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, India. e-mail: tapwalriya@kgpian.iitkgp.ac.in.com, (pallv.deb, sudim)@iitkgp.ac.in

S. K. Pal is with the Department of Mechanical Engineering, Indian Institute of Technology Kharagpur, India. e-mail:skpal@mech.iitkgp.ernet.in

together in both virtual BC and heterogeneous physical BCs.

Example Scenario: Consider a manufacturing industry that has heterogeneous branches spread across the world. These branches involve various activities based on the granular parameters. These activities need *managed execution, isolation, sharing of data, real-time computation, and efficient storage*. This mandates the need for isolated storage along with real-time computation. As shown in Fig. 1, *Shadows* allows these branches to store the data of each activity separately in different BC and provide a unified view of a single BC to the end-user. Further, if all the branches generate data concurrently, Shadows allows them to store data in a parallel manner. The end-user fetches the data of the industry, considering that there is only a single BC storing it. The parallel computation of consensus in Shadows also reduces the delay involved in storing the data.

Definition 1. ***Parallel Consensus:*** *Parallel consensus means reaching an agreement to add data in a parallel manner for all the blockchains without waiting for the resources. This helps in handling the data of all the activities involved in the industry in a parallel manner.*

Definition 2. ***Stable Systems:*** *In stable systems, all the virtual nodes are balanced. It means that each of the virtual nodes of the system is neither underutilized nor over-utilized.*

A. Motivation

Managing data from heterogeneous activities in industries is crucial, as it is useful for employing various activities such as tracking jobs, defect identification, quality assurance, and maintaining supplier relations. In order to maintain the integrity of data, there is a need for a solution that offers immutability and security. BC is a popular solution for storing data that can offer transparency, immutability, and security [3]. However, utilizing a single conventional BC (Etherium, Hyperledger, Bitcoin, and others) leads to a dormant consensus and inefficient data management. Apart from this, heterogeneous activities also require *managed execution, isolation, real-time processing, data sharing, and aggregation*, which a conventional BC cannot offer. In order to resolve this, the possible solution is to utilize multiple BCs. However, utilizing multiple BCs for handling the data of heterogeneous activities leads to the wastage of resources, which motivates us to virtualize the BC network. Also, to provide a cost and energy-efficient solution to manage the data of heterogeneous activities, there is a need to strategically virtualize the BC and store the data of these activities separately without wasting the resources. Apart from this, the need for security, data integrity, and the interaction between different BC motivates us to propose three different smart contracts which further empower *Shadows*.

B. Contribution

We virtualize the BC nodes for optimizing the resources, managing the data, and empowering the same by proposing smart contracts for authenticating users and avoiding data islands. To attain this, the explicit set of *contributions* of this work is as follows:

- **Shadows:** We virtualize the BC to achieve parallel consensus, isolation, sharing of data, and efficient utilization of resources. Apart from this, we store the data of heterogeneous activities in different virtualized BC, which results in distributed storage in addition to lower accessing time and better management of data.
- ***Smart Contracts to Empower Shadows:*** We propose three smart contracts: 1) Antumbra, 2) Penumbra, and 3) Umbra to empower *Shadows* and provide guided interoperability. The major functionalities of these smart contracts are as follows:
  - ***Antumbra:*** Antumbra provides the outer view of the VC and is responsible for authenticating the users for creating private BC.
  - ***Penumbra:*** Penumbra selects appropriate BC for the storage of data from heterogeneous activities and partially stores the data of industry over various BCs.
  - ***Umbra:*** Umbra is liable for the communication of BCs and provides a unified view of data to the end-users by hiding the inner storage information.
- ***Robustness:*** *Shadows* handles overloaded activities by dynamically allocating the resources to different activities depending upon their needs.
- ***Optimization:*** We utilize osmotic computing for allocating the resources among different blockchains.
- ***Evaluation:*** We discuss the feasibility of *Shadows* by performing an extensive experiment and deploying virtual machines on Raspberry Pis. We also demonstrate the advantages of utilizing *Shadows* over CBC.

The names used in this paper are shown in Table I.

Table I: Nomenclature table

Name	Description
BC	Blockchain
VC	Virtual Chain
Antumbra	Smart contract for authentication
Penumbra	Smart contract for selecting appropriate BC for storage
Umbra	Smart contract responsible for the communication of BC
VN	Virtual nodes
2FA	2 Factor Authentication
PoW	Proof-of-Work
CBC	Conventional BC
SBC	Separate BC

II. Related Work

A. Blockchain

Misra *et al.* [4] proposed a BC for IoT devices for implementing security and also proposed a clock mechanism for synchronizing the BC with non-real-time IoT devices. Further, Pathak *et al.* [5] utilized the concept of partial decentralized BC to

Table II: Difference of *Shadows* with similar BCs

Blockchain	Description	Virtualization of BC nodes	Interoperability	Parallel consensus	Resource wastage
Parallel Chain	Achieve interoperability among different network of BCs that configures all four modes (public, private, consortium and hybrid) of BC.	No	Yes	Yes	High
Multichain	Platform that establish private BCs and allow interactions between them.	No	Yes	No	High
Blockchain with virtual machine architecture	Creates BC over the virtual network.	Yes	No	No	High
Shadows (proposed)	Virtualize the network of BC and creates multiple BCs for different activities over the single network.	Yes	Yes	Yes	Low

provide transparency and security to Unmanned Arial Vehicles (UAV). They performed the virtualization of UAVs by utilizing the BC and provided services to various industrial applications by reliably transmitting the data. Similarly, Chang *et al.* [6] proposed SynergyChain for the reliable transmission of data by utilizing the data from multiple chains and organizing it in a single chain. Apart from data transmission, Aida *et al.* [7] utilized the multichain architecture for counterfeit detection, management of product life cycle, and tracking of products in the Agrifood industry. They created two virtual machines and enabled their interaction by realizing Multichain. Further, Guo *et al.* [8] proposed a BC based on the reputation value to optimize it and improve its efficiency. Further, Mirko *et al.* [9] proposed REchain based on Multichains to store metadata for real estate. They proposed immutable streams for publishing the purchase and sales offers by utilizing Multichain. Misra *et al.* [10] presented BC for Software Defined Network (SDN) to settle the flow rules of IoT devices as well as the fog nodes. Similarly, Nazarabadi *et al.* [11] proposed a light chain that provides addressable blocks for the easy accessing of data and results in lowers accessing time. Liu *et al.* [12] proposed a BC for the secret sharing of data by combining the same with the conventional Byzantine Fault Tolerant method. Similarly, Feng *et al.* [13] utilized the concept of device score to propose BAFL that ensures security and efficiency. Further, they used entropy to measure the quality of the federated learning model. Further, Liu *et al.* [14] proposed asynchronous and parallel smart contracts for distinguishing execution and consensus nodes for the parallel executions of the transactions. Zhang *et al.* [15] proposed a control system based on event triggered mechanism for providing security to the integrated energy systems (IES). Further, they also proposed event triggered communications between the IES which resulted in low cost. Jiang *et al.* [16] proposed a spectrum acquisition system based on BC which resulted in minimizing the transmission power while satisfying the threshold values of transmission. There are various parameters which needs to be handled, The proposed system proves to be beneficial for storing these parameters. Fu *et al.* [17] proposed a BC based network function virtualization framework to ensure security and solve the problem of trust. In this work there are different services which are difficult to handle. In order to resolve this issue there is a need of solution which can handle these services in parallel fashion.

B. Virtualization

Belt *et al.* [20] surveyed the virtualization in the wireless networks and studied various resource(storage, memory, computing power, and other) allocation schemes among various virtual nodes to improve the scalability and enhance parallel processing. In order to utilize the computing resources efficiently in the IoT devices, Ogawa *et al.* [21] introduced virtualization of IoT devices and proposed resource scaling as well as microservice scaling by utilizing Docker, Kubernetes, and Apache Kafka. Further, Cheg *et al.* [22] also proposed wireless virtualization for IoT devices to harvest energy and designed virtual resource mapping through alternate iterations of the Stackelberg game.

C. Resource Allocation

Bahreini *et al.* [23] addressed the resource allocation problem in mobile devices and proposed auction-based resource allocation methods that fulfill the heterogeneous demands of various systems. Further, Wang *et al.* [24] proposed self-adaptive resources management framework by calculating the Quality of Service (QoS) value of the current workload and utilizing the same for predicting the future QoS value and allocating resources efficiently using PSO based run-time decision algorithm. Battula *et al.* [25] proposed a model which predicts the future availability of the resources and allocates these in a time-sensitive manner. These proposals did not consider the energy consumption for resource allocation. To prevent high energy consumption, Than *et al.* [26] proposed resource allocation algorithms by considering various factors such as power management techniques, power models, and resource allocation policies to evaluate the real-time workload and further allocated the resources accordingly. Further, Gamal *et al.* [19] utilized the concept of osmotic computing to achieve resource allocation and load balancing. They proposed a hybrid meta-heuristic technique that automatically deploys virtual machines by integrating a bio-inspired load balancing algorithm with the osmotic behavior.

D. Synthesis

Blockchain, when applied to heterogeneous activities in industries, suffers from various challenges such as storing heterogeneous data, achieving parallel consensus, and efficient

Table III: Difference of *Shadows* with some existing works

Paper	Communication between BC	Migration time	Load for migration	Virtualization of BC	Multichains
Chang *et al.* [6]	✓	×	×	×	✓
Aida *et al.* [7]	×	×	×	×	✓
Ahmad *et al.* [18]	×	×	×	×	✓
Mirko *et al.* [9]	×	×	×	×	✓
Gamal *et al.* [19]	×	✓	×	×	×
Shadows (proposed)	✓	✓	✓	✓	✓

management of data. The data from various activities are subject to non-uniformity in terms of volume, data type, sampling rate, variability, and others. Conventional BC deployments in literature do not consider the efficient utilization of resources as well as real-time data storing and accessing, as shown in Table II. To achieve this, we virtualize the nodes of the BC network and deploy Multichains for different activities. Also, as shown in Table III, the existing Multichains do not virtualize the node of the BC network for efficient utilization of resources and consider the communication between BC as well as migration of virtual nodes (VN) (considering both total migration time and the load of the migrating VN). To resolve these issues, we propose *Shadows* (virtualized BC) utilizing smart contracts for the efficient utilization of the resources as well as achieving parallel consensus.

III. System Model

A. Network Architecture

As shown in Fig. 1, we consider a set of activities $A = \{a_1, a_2, a_3, \ldots, a_n\}$, where each activity consist of set of sensors $S = \{s_1, s_2, s_3, \ldots, s_m\}$ which sense the data produced by various devices indulge in these activities. The sensors forward the data to the *Shadows*, which stores the data of different activities in different BC. *Shadows* consists of three smart contracts which handle the data of different activities: 1.) Antumbra, 2.) Penumbra, 3.) Umbra and avoid the wastage of various resources. On the activation of new activity in the industry, Antumbra creates new BC and dynamically allocates resources based on the demand of the activity. For the creation of a new BC, Antumbra virtualizes the nodes of the BC network and allocates these virtual nodes to different activities for the creation of various BC. The virtual nodes associated with different activities achieve consensus on the creation of data and provide storage to the same. Further, Antumbra utilizes osmotic computing to dynamically allocate resources among the virtual nodes based on the load of the activities. Moreover, Penumbra selects the appropriate BC for the storage of data of various activities, and Umbra is responsible for the communication of various BCs to provide a unified view of a single BC to the end-users. Umbra utilizes the addresses of the genesis blocks of different BCs to enable communication between these and avoid data islands. The end-users request data from the *Shadows*, considering a single BC, and fetches the data of various activities on-demand.

B. Shadows

We consider various types of independent activities utilizing IIoT devices and producing heterogeneous data. To provide an independent working environment in a cost-effective manner, we propose *Shadows*, which consists of separate BC for storage and achieves parallel consensus for each BC. To achieve this, we virtualize the nodes of the BC network and map these virtual nodes to different activities for achieving consensus and storing their data. We also calculate the optimum virtualization of a node possible which can handle data as well as achieve consensus. Further, to aid the virtualization of the BC network, we propose three smart contracts whose details are as below:

- ***Antumbra:*** This smart contract provides an outer view of the VC and is responsible for the authentication of the users, who can create more virtual nodes on the activation of new activity in the industry. We utilize two-factor authentication (2FA) for the authentication of valid users. In 2FA, the user signs up to the application site using his *username* and *password*, which further generates a token for the corresponding user. Further, the user logs in to the smart contract using *username*, *password*, and *token*, which authenticates the user and allows the same to make modifications in the network.
- ***Penumbra:*** This smart contract is responsible for the allocation of virtual nodes to various activities for achieving consensus and storing data in their respective BC. To achieve this, Penumbra utilizes the identities of the activities and virtual nodes to assign the data of a particular activity to the respective virtual BC network.
- ***Umbra:*** Umbra is responsible for the interaction of various BC and provides a unified view to the end-users. Umbra achieves the same by storing the address of the genesis block of each BC and fetching the data by using the same. This also avoids data islands as one BC also fetches the data of the other by utilizing the genesis' address stored in the Umbra as well as the timestamp of the data which is required. As shown in Fig. 2, the end-users log in using their Id and password, and Umbra verifies the same. Umbra denies access if the Id or the password does not match with the already registered IDs and corresponding passwords. However, if the Id and password match, Umbra generates the key using the same and fetches all the hashes associated with

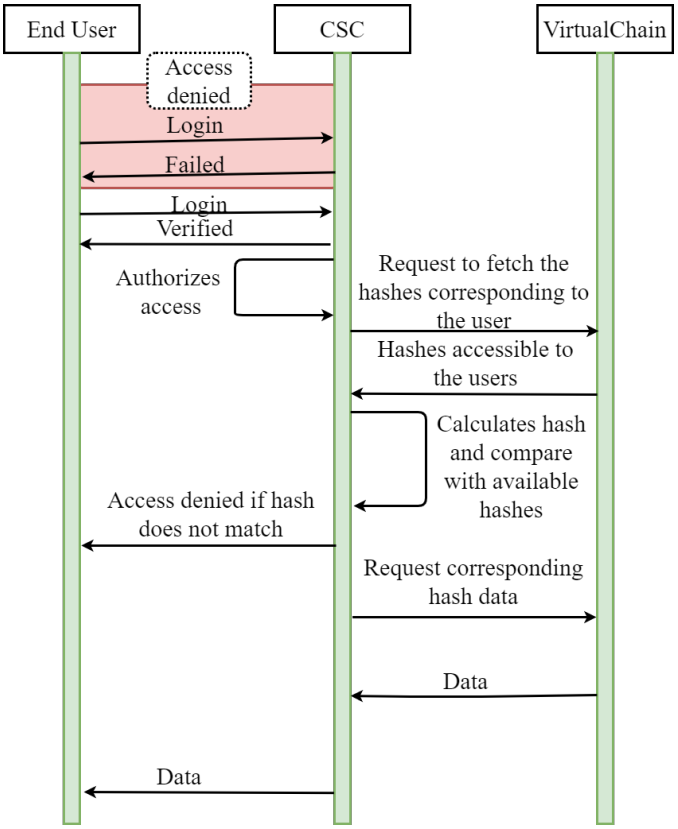


Figure 2: Process of access authorization done by Umbra

the key that is accessible to the corresponding user. It further generates the hash of the requested data using the requested timestamp and activity id. If the generated hash matches with one of the accessible hashes (hash values that are accessible to particular users), Umbra requests the corresponding data from the respective BC by utilizing the stored addresses and presents the data to the end-user. If the generated hash does not match with the accessible hashes, Umbra denies the request of the end-users.

C. Virtualizing Blockchain (VirtualChain)

Shadows pools various BCs storing the data of heterogeneous activities and makes it appear to be a single BC handling the heterogeneous data and facilitating efficient management as well as monitoring of the same. To achieve this, we create separate BCs for storing the data of different activities and providing encapsulation to the same. The BC stores the data of different activities at varying mining time according to their needs and provide separate storage for each activity. Further, to provide a unified view to the end-users of having only single storage, we create a virtualization layer consisting of the smart contract (Umbra), which makes the communication of BC possible in addition to the accessing of data efficiently. As shown in Fig. 3, in physical, we have separate BCs for storing the data of each activity; however, the virtualization layer provides a unified view of single storage (VirtualChain) to the end-user.

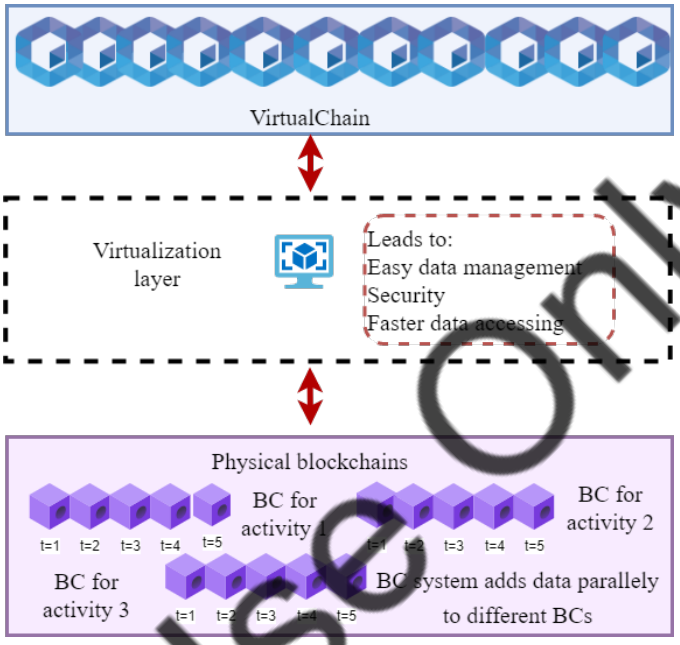


Figure 3: Advantages of blockchain virtualization

D. Resource Allocation

We consider a set of BC nodes, $B = \{b_1, b_2, b_3, \ldots, b_k\}$ and a set of virtual nodes, $VN = \{v_1, v_2, v_3, \ldots, v_p\}$, where k is the number of nodes present in the BC network and p is the total number of virtual nodes. Further, based on the requirement of the activities, we assign a varying number of virtual nodes to each and deploy these nodes on different BC nodes. The deployment of virtual nodes to the BC network nodes process the different activities in a parallel manner and utilizes the concept of osmotic computing for efficient utilization of resources.

1) Assignment of virtual nodes: We consider p virtual nodes and n heterogeneous activities, where we assign these virtual nodes to different activities according to their CPU and memory requirements. The CPU and memory requirement of i^{th} activity (α_i) is $\mu_{CPU} + \mu_{mem}$ and the number of virtual nodes assigned to each activity is given as:

$$\eta_i = \frac{p \times \alpha_i}{\sum_{i=1}^{n} \alpha_i} \tag{1}$$

2) Load balancing: We calculate the load of each node of BC network by calculating the loads of virtual nodes deployed on each and categorize these as *over-utilized*, *under-utilized*, and *balanced*. We calculate the average load of i^{th} BC node as $L_i = (\sum_{i=1}^{p} V_{ijr})/p$, where V_{ijr} is the load of i^{th} virtual node on j^{th} BC node due to r^{th} activity and is given as $\mu_{CPU_{ijr}} + \mu_{mem_{ijr}}$. Further, the load of i^{th} BC node is given as the standard deviation of average load of i^{th} BC node as well as the average load on all the BC nodes ($L = (\sum_{i=1}^{k} L_i)/k$) and

Algorithm 1: Algorithm for categorizing the nodes.

Inputs: CPU and memory requirements of each virtual node (V_{ijr}), number of BC nodes (k), and Number of virtual nodes (p)
Outputs: Load of BC nodes
Initialize:
L=0
Procedure:
for *each BC node in the network* **do**
  $L_i = 0$
  for *each virtual node in the BC node* **do**
    if *virtual node belongs to corresponding BC node* **then**
      $L_i = L_i + \frac{V_{ijr}}{p}$
    end
    // Average load on v_j
  end
end
for *each BC node in the network* **do**
  $L = L + \frac{L_i}{k}$ // Standard deviation of load on BC nodes
end
for *each BC node in the network* **do**
  $\sigma_i = \sqrt{(\frac{\sum_{i=1}^{k}(L-L_i)^2}{k})}$
  if $\sigma_i \geq L$ **then**
    over-utilized
  end
  // L is the upper threshold
  else if $\sigma_i \leq min(L_i)$ **then**
    under-utilized
  end
  // $min(L_i)$ is the lower threshold
  else
    Balanced
  end
end

is given as:

$$\sigma_i = \sqrt{\frac{1}{k} \times \left(\sum_{i=1}^{k} \left(\underbrace{\left(\frac{\sum_{i=1}^{k} L_i}{k} \right)}_{L} - \underbrace{\left(\frac{\sum_{i=1}^{p} V_{ijr}}{p} \right)}_{L_i} \right)^2 \right)} \tag{2}$$

We compare the Equation 2 with L (upper threshold) and $min(L_i)$ (lower threshold) and categorize b_i (i^{th} BC node) as over-utilized if its value is greater than the value of L and under-utilized if its value is less than the value of $min(L_i)$ as shown in Algorithm 1.

3) Selecting optimal virtual node for migration: After finding the over-utilized virtual nodes, we find the most suitable virtual node for the migration. We consider the total migration time (t_m) as well as the load of the virtual node on that physical BC node (V_{ijr}) to select the virtual node for migration. We not only give preference to the node with the shortest t_m; however, we also consider the load V_{ijr} of the heavy virtual node for migration and migrate the node with the lowest time-load ratio (τ_r). This results in a more stable system of BC nodes as this technique equally divides the load of the virtual nodes among the physical BC nodes. Further, the τ_r is $(((V_i - V_u)/\beta) + (V_u/\beta)) \times (\beta/V_u)$, where V_i is the allocated memory to the i^{th} virtual node, V_u is the utilized memory by the i^{th} virtual node and β is the allocated bandwidth to that node for migration. Specifically, we migrate the node with lowest τ_r as shown in Algorithm 2, and it is given as:

$$\tau_r = \frac{V_i}{V_u} \tag{3}$$

Algorithm 2: Algorithm for selecting optimal virtual node for migration.

Inputs: Over-utilized BC nodes B_{over}
Output: Optimal virtual node for migration.
Procedure:
for *each node in* B_u **do**
  for *each j in VN* **do**
    if $V_j \in B_i$ **then**
      $\tau_r = \frac{V_i}{V_u}$
    end
  end
  Migrate the node with min (τ_r);
end

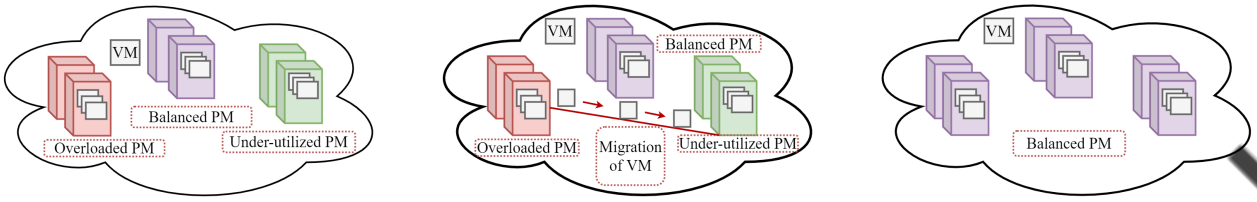
4) Selecting optimal BC node for placement: We calculate the fitness (ϕ) of each migrating virtual node with the under-utilized BC node and deploy the virtual node over the BC node with maximum value of ϕ as shown in Algorithm 3. The fitness of the migrating node is calculated as:

$$\phi_{V_m,B_u} = \frac{B_u^{CPU} - V_m^{CPU}}{V_m^{CPU}} + \frac{B_u^{mem} - V_m^{mem}}{V_m^{mem}} \tag{4}$$

In equation 4, ϕ_{V_m,B_u} is the fitness of virtual migrating node (V_m) in the BC node (B_u), which is under-utilized. Further, B_u^{CPU} and B_u^{mem} represent the amount of CPU and memory the BC node have whereas V_m^{CPU} and V_m^{mem} represent the requirement of the virtual node.

As shown in Fig. 4, initially, an unbalanced system consists of over-utilized B, balanced B, and under-utilized B (Fig. 4(a)). Further, by utilizing Algorithm 1, we calculate the over-utilized and under-utilized B in addition to the optimal migrating VN by utilizing Algorithm 2. Further, we find the optimal B by utilizing Algorithm 3 for the placement of migrating VN and migrate the same as shown in Fig. 4(b) to achieve a balanced system as shown in Fig. 4(c)

5) Complexity: The time complexity for categorizing the nodes into over-utilized, balanced, and under-utilized is $O(k)$ where k is the number of nodes present in the BC network. This is because we have to iterate through all the nodes only once to check their load. Further, we select the appropriate



(a) Unbalanced system (b) Migration of virtual nodes (c) Balanced system

Figure 4: Load balancing by utilizing osmotic computing in Shadows

Algorithm 3: Algorithm for selecting optimal BC node for placement.

Inputs: Set of under-utilized BC nodes B_u and set of migrating virtual nodes V_m
Output: Optimal BC node for virtual node placement.
Procedure:
for `each i in` V_m **do**
  for `each j in` B_u **do**
    $\phi_{V_m,B_u} = \frac{B_u^{CPU} - V_m^{CPU}}{V_m^{CPU}} + \frac{B_u^{mem} - V_m^{mem}}{V_m^{mem}}$
  end
  $\nu = max(\phi_{V_m,B_u})$;
  Assign i^{th} node to B_ν;
end

migrating node from the over-utilized B in $O(kl)$ where l is the number of virtual nodes present at each B. We attribute this complexity to iterate through all the VN present at all the over-utilized B and place it at suitable B in $O(kl)$. The overall complexity is $O(k + kl)$.

Proposition 1. *Irrespective of a resource-constrained environment, Shadows is an optimal solution for heterogeneous activities.*

Proof. Typically, industries deal with a heterogeneous set of activities $A = \{a_1, a_2, a_3, \ldots, a_n\}$ consisting of myriad range of sensors $S = \{s_1, s_2, s_3, \ldots, s_m\}$. Handling these data in a single BC is challenging due to the varying attributes and applications. Further, the utilization of Multichains for handling data of different activities is not suitable as this results in the wastage of resources. In *Shadows*, we virtualize the nodes of the BC network to handle multiple BC parallelly and utilize the resources efficiently by balancing the load using osmotic computing, which reduces the wastage of resources and makes *Shadows* suitable for the resource-constrained environment. □

Proposition 2. *Shadows results in a stable system.*

Proof. Shadows utilizes both the load produced by the VN on the B and the total migration time to find the optimal migrating VN. This results in finding the VN, which not only results in the least migration time but also balances the load of the B and results in a more balanced system. Further, the system is stable only if $\phi_{V_m,B_u} > 0$ as it represents the placement of VN on the underutilized BC node. In our system, we place the VN on B if $B_u^{CPU} >> V_m^{CPU}$ and $B_u^{mem} >> V_m^{mem}$. We utilize Equation 4 to prove the stability of our system and obtain:

$$\frac{B_u^{CPU} - V_m^{CPU}}{V_m^{CPU}} + \frac{B_u^{mem} - V_m^{mem}}{V_m^{mem}} > 0 \quad (5)$$

$$\implies \phi_{V_m,B_u} > 0 \quad (6)$$

From Equation 6, we conclude that *Shadows* results in stable system.

□

Proposition 3. *Shadows results in managed execution and isolation of data.*

Proof. Shadows utilizes a set of blockchains $BC = \{b_1, b_2, b_3, \ldots, b_m\}$ for storing the data of different activities $A = \{a_1, a_2, a_3, \ldots, a_n\}$ separately such that $m = n$ and $a_i \longleftrightarrow bi$. We will prove this by contradiction method and assume that there are two activities a_i and a_j which are not isolated. As given, $a_i \longleftrightarrow b_i$ which implies that $m \neq n$. However, $m = n$ which proves that a_i and a_j are isolated. □

Proposition 4. *Irrespective of distributed storage of data, Shadows provide an unified view to the end-users (aggregation).*

Proof. Shadows stores the data of different activities $A = \{a_1, a_2, a_3, \ldots, a_n\}$ on separate $BC = \{b_1, b_2, b_3, \ldots, b_m\}$; however, as mentioned in Section III, Umbra makes the communication between different BC possible by storing the authorization access of different users and fetching data from various BCs on their request. The end-users accesses the data of various activities using Umbra which hides the communication details of various BCs and provide unified view to the end-users. □

IV. Performance Evaluation

A. Experiment Setup

We use Python 3 to execute our routines and utilize the data of 3 different activities as a proof of concept. Activity 1 consists of the data of digital twin, Activity 2 consists of the data of

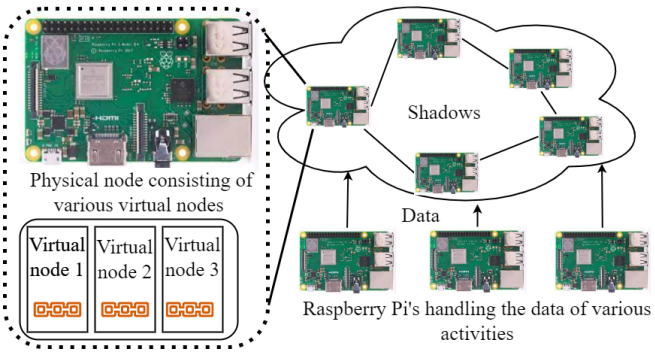


Figure 5: Experimental setup for Shadows

friction stir welding, and Activity 3 consists of the machine health data. Further, we use a virtualized BC, which stores the data of different activities in different BC (refer Fig. 5) and utilizes smart contracts for selecting the appropriate BC as well as communicating the same. For achieving the same, we virtualize the Raspberry Pis in the BC network and store the data on respective virtual nodes. Further, we store the smart contracts on different physical BCs. These smart contracts are broadcasted and executed by all the physical nodes in the BC network in order to record the latest data.

B. Deployment Architecture

We create a network of Raspberry Pis, which continuously send data to the *Shadows*. In the *Shadows*, we virtualize the nodes of the BC network and allocate these virtual nodes to various activities, as shown in Fig. 5. We allocate the data of the particular activity to the respective BC by utilizing the identity of the activity. Further, at the request of the end-users, we identify the appropriate BC consisting of the requested data and access that by utilizing the address of the genesis block.

C. Benchmark Schemes

Alzahrani *et al.* [27] used a conventional blockchain (CBC) for storing the data produced by various sensors in industries and used the same in monitoring the industrial condition remotely. We compare the storage complexity and the resource utilization of *Shadows* with CBC to prove its feasibility. Further, Aida *et al.* [7] used separate blockchains (SBC) for storing the data of different activities and used that data for tracking the products. We also compare *Shadows* with SBC to show the expediency of *Shadows* in preventing resource wastage.

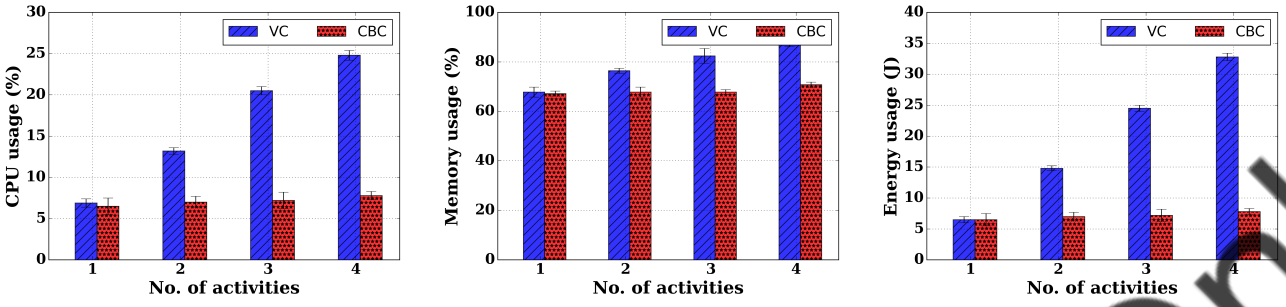
D. Results

1) CPU Utilization: We observe the CPU utilization while executing VC and compare them with the utilization of separate BC for different activities. We perform 30 iterations to record our observations and observe that VC utilizes CPU more efficiently as compared to the conventional BC (refer Fig. 6(a)). VC shows an increase of 18% in CPU usage for four activities. We also observe that with the increase in activities, the CPU utilization also increases. We attribute this increase to the distribution of processing power among various virtual nodes (VN). In particular, for a single activity, the CPU utilization for both the VC and conventional BC is comparable and is 6%. This is because, in VC, for a single activity, we dedicate the whole processing power to that activity only. Further, there is an increase of 8%, 15%, and 18% CPU utilization for 2, 3, and 4 activities, respectively, as compared to conventional BC. We infer from our observation that, with the increase in virtualization, the utilization of the CPU also increases. Apart from this, virtualization depends upon the processing power required by an activity.

2) Memory Utilization: We record the memory consumption while executing VC and observe that it utilizes 92% of the memory for executing four activities (refer Fig. 6(b)). We record our observations by performing 40 iterations and observe that VC utilizes 65%, 78%, and 81% for 1, 2, and 3 activities, respectively. This behavior is because VC distributes its memory among various virtual nodes with the increase in the number of activities. More number of activities require more memory for their execution which improves memory utilization. In general, the memory requirement for a single activity in conventional BC and VC are comparable, and there is only 1% increase in the VC. We attribute this increase to the execution of various smart contracts (Antumbra, Penumbra, and Umbra). Further, there is 13%, 19%, and 27% increase in the memory as compared to the conventional BC for 2, 3, and 4 activities, respectively. We entail from our observations that the virtualization of BC nodes into a suitable number of VN increases the utilization of memory more efficiently.

3) Energy Utilization: We observe the energy utilization while executing VC and compare it with the utilization of separate BC for different activities. We perform 30 iterations to record our observations and observe that VC utilizes more energy as compared to the conventional BC (refer Fig. 6(c)). VC shows an increase of $26J$ in energy utilization for four activities. We also observe that with the increase in activities, the requirement for energy also increases. We attribute this increase to the consumption of processing power among various VN. In particular, for a single activity, the energy utilization for both the VC and CBC is comparable and is $6J$. This is because, for a single activity, both VC and CBC require comparable processing power. Further, there is an increase of $9J$, $18J$, and $26J$ energy for 2, 3, and 4 activities, respectively, compared to CBC. We infer from our observation that, with the increase in virtualization, the utilization of energy also increases.

4) Response Time: We observe the response time for executing VC and compare it with the response time of separate BC (SBC) for different activities as well as single CBC. We perform 30 iterations to record our observations and observe that VC requires less response time as compared to the single CBC (refer Fig. 7(a)). VC shows a decrease of 38% in response time for four activities compared with a single CBC. We also observe that VC requires 0.7 sec (average) more as compared to the separate BC for different activities. We attribute this increase to the execution routines of various smart contracts (Antumbra, Penumbra, and Umbra). In particular, for a single activity, the response time for the VC, separate BC,

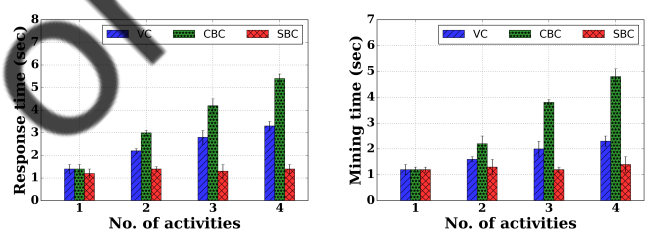


(a) CPU usage for adding data to BC (b) Memory usage for adding data to BC (c) Energy usage for adding data to BC

Figure 6: Resource usage for adding data to VC and CBC

and single CBC are comparable and is 1.2 sec. This is because, for a single activity, we dedicated all the resources to that activity only. Further, there is a decrease of 31%, 47%, and 56% response time for 2, 3, and 4 activities, respectively, as compared to a single CBC. Apart from this, there is 1.125 sec (average) diminutive increase in response time when compared to separate BC for different activities, which is acceptable. We infer from our observation that response time decreases when we provide separate resources for different activities.

5) Mining Time: We observe the mining time for executing VC and compare it with the mining time of SBC for different activities as well as a single CBC. We perform 30 iterations to record our observations and observe that VC requires less mining time as compared to the single CBC (refer Fig. 7(b)). VC shows a decrease of 56% in mining time for four activities when compared with a single conventional BC. We also observe that VC requires 0.9 sec more as compared to the SBC for different activities. We attribute this increase to the execution routines of various smart contracts (Antumbra, Penumbra, and Umbra). In particular, for a single activity, the mining time for the *Shadows* (VC), SBC, and single CBC is comparable and is 1 sec. This is because, for a single activity, we dedicated all the resources to that activity only. Further, there is a decrease of 30%, 31%, and 40% mining time for 2, 3, and 4 activities, respectively, as compared to a single conventional BC. Apart from this, there is 0.9 sec (average) diminutive increase in mining time when compared to SBC for different activities, which is acceptable. We infer from our observation that mining time decreases when we provide separate resources for different activities.



(a) Response time for accessing data from BC (b) Mining time for adding data to BC

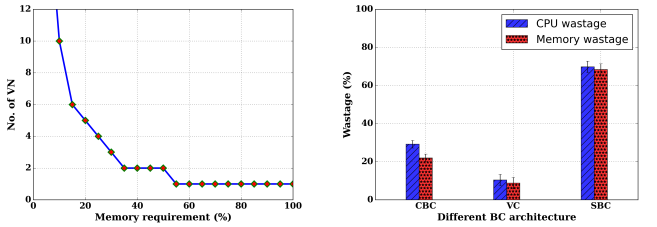
Figure 7: Response and mining time of data for different BC

6) CPU Wastage: We observe the wastage of CPU while executing VC and compare them with the CPU wastage in the case of SBC as well as common BC for different activities. We perform 30 iterations to record our observations and observe that the CPU wastage in VC is the least (refer Fig. 8(b)). VC shows a decrease of 55% in CPU wastage when compared with the wastage done by SBC. We attribute this decrease to the consumption of CPU by various VN. In particular, the CPU wastage done by VC, common BC for all the activities, and SBC for different activities is 15%, 35%, and 70%. We infer from our observation that the wastage of processing power decreases with the increase in virtualization.

7) Memory Wastage: We observe the wastage of memory while executing VC and compare them with the case of SBC as well as common BC for different activities. We perform 30 iterations to record our observations and observe that the memory wastage in VC is the least (refer Fig. 8(b)). VC shows a decrease of 60% in memory wastage when compared with the wastage done by SBC. We attribute this decrease to the consumption of memory by various VN. In particular, the wastage done by VC, common BC for all the activities, and separate BC for different activities is 10%, 25%, and 70%. We infer from our observation that, with the increase in virtualization, the wastage of memory decreases.

8) Virtual Node Requirement for Varying Memory Requirement: We observe the number of virtual nodes possible with varying memory requirements and perform 30 iterations to record our observations. We assume that each virtual node requires at least 50% of the memory and observe that the number of VN on a B depends upon the memory requirement of the VN. In particular, we observe that we can deploy 10 VN when the requirement of each VN is 5% as shown in Fig. 8(a). We attribute this behavior to the less resource utilization of each VN. We entail from our observation that the number of possible deployment of VN decreases with the increase in memory requirement of each VN.

9) Fitness Against Required Memory by Virtual Nodes: We observe the fitness of various VN at a particular B while migrating these from source B to the suitable destination B. We perform 40 iterations while migrating VN and observe that the fitness value depends upon the memory requirement of the migrating VN. We observe that the fitness decreases as there is an increase in the memory required by the VN (refer Fig.

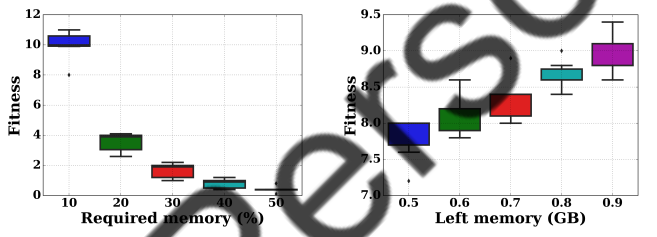


(a) Virtual nodes' requirements for varying memory requirements (b) Resource wastage by different BC networks

Figure 8: Virtual node requirements and resource wastage

9(a)). This behavior is because the heavy VN generates load on destination B and further results in an unbalanced system. In particular, there is 66% decrease in the fitness when the memory requirement of the VN varies from 10% to 50%. We entail from our observations that fitness decreases with the increase in the memory requirement of the VN.

10) Fitness Against Available Memory of BC Nodes: We observe the fitness of a particular VN at various BC nodes(B) while migrating that from source B to the suitable destination B. We perform 40 iterations while migrating VN and observe that the fitness value depends on destination B's memory. We observe that the fitness increases as there is an increase in the memory required by the VN (refer Fig. 9(b)). This behavior is because there are fewer chances of over-utilizing at destination B if the available memory is more. In particular, there is 18% increase in the fitness when the memory available at the B varies from 0.5 GB to 0.9 GB. We entail from our observations that fitness increases with the increase in the availability of memory at the B.

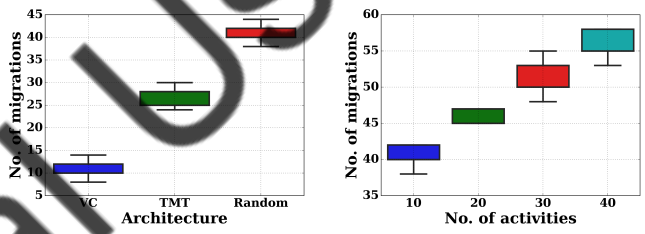


(a) Fitness with required memory of virtual node (b) Fitness with left memory of BC node

Figure 9: Fitness for migrating virtual nodes

11) No. of Migrations for Different Architectures: We observe the number of migrations of VN while executing VCand compare the same with the number of migrations required while considering only total migration time (TMT) and the random migration. We perform 40 iterations to present our observations and observe that the migrations of V required by VC are the least. In particular, VC requires 15 migrations for four activities, whereas TMT and random architecture require 30 and 45 migrations, respectively, as shown in Fig 10(a). The reason behind this behavior is that we consider the load of the VN along with the total migration time to select the suitable B. We also observe that there is 67% decrease in the number of migrations while we utilize VC as compared to random migrations of VN. We infer from our observations that considering the load and total migration time makes the system more stable and reduces the number of migrations.

12) No. of Migrations for Varying Number of Activities: We observe the number of migrations of VN for the varying number of activities and perform 40 iterations to present our observations. We observe that the migrations of VN increase with the increase in the number of activities. In particular, VC requires 57 migrations for 40 activities, as shown in Fig 10(b). Further, the number of migrations required by 10, 20, and 30 activities are 43, 48, and 55, respectively. The reason behind this behavior is that more number activities create an imbalance in the system due to their increasing demand for memory and CPU. We also observe that there is a 34% increase in the number of migrations while increasing the number of activities from 10 to 40. We infer from our observations that the number of migrations increases with the increase in the number of activities.

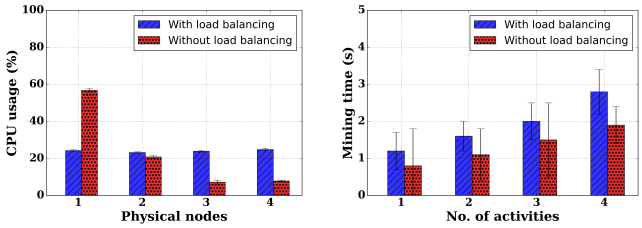


(a) No. of migrations for different BC architecture (b) No. of migrations for different activities

Figure 10: No. of migrations required to make stable system

13) Performance overhead of load balancing: For observing the performance overhead of load balancing in Shadows. We calculate the CPU usage by different physical nodes in the BC network as well as the mining time of the BC network (with or with load balancing) for the different number of activities. We perform 40 iterations of each experiment to record our observations. From Fig. 11(a), we observe that the CPU usage of each physical node is almost equivalent while utilizing load balancing. However, without load balancing, the tasks are unevenly distributed among various nodes, which is represented by the CPU usage of all the physical nodes. On average, the maximum difference between the CPU usage is 50%. The reason behind these observations is that with load balancing tasks are evenly distributed among various nodes, which consequently results in almost equal CPU usage at all the physical nodes. Apart from this, the mining time with load balancing is more compared to without load balancing. Specifically, there is 44.44% increase in mining time as shown in Fig. 11(b). This is due to the time taken for load balancing. From our observations, we imply that the load balancing adds performance overhead to Shadows; however, it results in the stability of the system by distributing the tasks evenly over the physical nodes.

In summary, compared to conventional BC practices, *Shadows* significantly reduces resource wastage, mining time, response time, and the number of VN's migrations, making



(a) CPU usage overhead (b) Mining time overhead

Figure 11: Performance overhead due to load balancing

it suitable for real-time applications with negligible energy overheads. From our observation, we comment that it is best suitable for scenarios that satisfy the following conditions:

- C1: Resource-constrained environment: From Section IV-D7 and IV-D6, we observed that *Shadows* reduces the wastage of resources as compared to the conventional BC deployments. This makes it suitable for the environment with a limited number of resources.
- C2: Real-time applications: From Section IV-D5 and IV-D4, we observed that *Shadows* significantly reduces the mining time and response time compared to the conventional BC deployments, which makes it suitable for real-time applications.
- C3: Energy-constrained devices: From Section IV-D3 and IV-D12, we observed a diminutive increase in energy as well as a significant decrease in the number of VN's migration as compared to the convention blockchain, which makes it suitable for the use in any legacy infrastructure with energy-constrained devices.

E. Discussions and Limitations

We propose a method –*Shadows*– to virtualize the nodes of the BC network in order to reduce the mining and response time in addition to the reduction of wastage of resources. Further, to balance the load among the virtualized nodes (VN), we utilize osmotic computing, which is responsible for the movement of V from the over-utilized BC nodes to the under-utilized BC nodes. To examine the effectiveness of the osmotic computing, we calculate the number of migrations required to make the system balanced and observed that the proposed method requires the least number of migrations as compared to the method utilizing only total migration time and the random technique for migration. Further, we aid *Shadows* by proposing three smart contracts which are responsible for the authentication, creation of virtual nodes, and the communication of different BC.

The limitation of the proposed method is that it does not consider the tamper-proof communication between various BC and implements the proposed solution over private BCs. However, the solution is also suitable for public BCs, and we plan to address the same in our extended work.

V. Conclusion

In this paper, we proposed a virtual BC (*Shadows*) to effectively utilize the resources in achieving real-time parallel consensus. The proposed method virtualized the nodes of the BC network and used these for achieving consensus in real-time. Further, to endow *Shadows*, we also proposed three smart contracts: 1) Antumbra (to authenticate the legitimate users), 2) Penumbra (to select the appropriate BC), and 3) Umbra (to make communication between BC possible and provide a unified view to the end-users). We aimed to compute the consensus in real-time by utilizing the resources efficiently and distributing heterogeneous data over multiple BCs for better management. We also performed extensive lab experiments to demonstrate the feasibility of *Shadows* in achieving the above-mentioned goals.

In this paper, we ceased our research to utilize the resources efficiently for computing parallel consensus and did not consider the tamper-proof communication between different BCs. We plan to address secure communication between the BC in our extended work and also implement the proposed solution over public BCs.

References

[1] Z. Zhou, B. Wang, Y. Guo, and Y. Zhang, "Blockchain And Computational Intelligence Inspired Incentive-Compatible Demand Response In Internet Of Electric Vehicles," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 3, no. 3, pp. 205–216, 2019.

[2] M. Maksimović, "The Role of Osmotic Computing in Internet of Things," in *Proceedings of the* 17^{th} *International Symposium Infotech Jahorina (INFOTEH)*, 2018, pp. 1–4.

[3] N. Mohamed and J. Al-Jaroodi, "Applying Blockchain in Industry 4.0 Applications," in *IEEE* 9^{th} *Annual Computing and Communication Workshop and Conference (CCWC)*, 2019, pp. 0852–0858.

[4] S. Misra, A. Mukherjee, A. Roy, N. Saurabh, Y. Rahulamathavan, and M. Rajarajan, "Blockchain at the Edge: Performance of Resource-Constrained IoT Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 1, pp. 174–183, 2021.

[5] N. Pathak, A. Mukherjee, and S. Misra, "AerialBlocks: Blockchain-Enabled UAV Virtualization for Industrial IoT," *Internet of Things Magazine*, vol. 4, no. 1, pp. 72–77, 2021.

[6] J. Chang, J. Ni, J. Xiao, X. Dai, and H. Jin, "SynergyChain: A Multichain-Based Data Sharing Framework with Hierarchical Access Control," *IEEE Internet of Things Journal*, pp. 1–1, 2021.

[7] A. Ismailisufi, T. Popović, N. Gligorić, S. Radonjic, and S. Šandi, "A Private Blockchain Implementation Using Multichain Open Source Platform," in *Proceedings of the* 24^{th} *International Conference on Information Technology (IT)*, 2020, pp. 1–4.

[8] S. Guo, Y. Qi, Y. Jin, W. Li, X. Qiu, and L. Meng, "Endogenous Trusted DRL-Based Service Function Chain Orchestration for IoT," *IEEE Transactions on Computers*, pp. 1–1, 2021.

[9] M. Avantaggiato and P. Gallo, "Challenges and Opportunities using MultiChain for Real Estate," in *Proceedings of the International Black Sea Conference on Communications and Networking (BlackSeaCom)*, 2019, pp. 1–5.

[10] S. Misra, P. K. Deb, N. Pathak, and A. Mukherjee, "Blockchain-Enabled SDN for Securing Fog-Based Resource-Constrained IoT," in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2020, pp. 490–495.

[11] Y. Hassanzadeh-Nazarabadi, A. Küpçü, and Özkasap, "LightChain: Scalable DHT-Based Blockchain," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 10, pp. 2582–2593, 2021.

[12] J. Liu, W. Li, G. O. Karame, and N. Asokan, "Scalable Byzantine Consensus via Hardware-Assisted Secret Sharing," *IEEE Transactions on Computers*, vol. 68, no. 1, pp. 139–151, 2019.

[13] L. Feng, Y. Zhao, S. Guo, X. Qiu, W. Li, and P. Yu, "Blockchain-based Asynchronous Federated Learning for Internet of Things," *IEEE Transactions on Computers*, pp. 1–1, 2021.

[14] J. Liu, P. Li, R. Cheng, N. Asokan, and D. Song, "Parallel and Asynchronous Smart Contract Execution," *IEEE Transactions on Parallel and Distributed Systems*, pp. 1–1, 2021.

[15] N. Zhang, Q. Sun, L. Yang, and Y. Li, "Event-Triggered Distributed Hybrid Control Scheme for the Integrated Energy System," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 2, pp. 835–846, 2022.

[16] M. Jiang, Y. Li, Q. Zhang, G. Zhang, and J. Qin, "Decentralized Blockchain-Based Dynamic Spectrum Acquisition for Wireless Downlink Communications," *IEEE Transactions on Signal Processing*, vol. 69, pp. 986–997, 2021.

[17] X. Fu, F. R. Yu, J. Wang, Q. Qi, and J. Liao, "Performance Optimization for Blockchain-Enabled Distributed Network Function Virtualization Management and Orchestration," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 6, pp. 6670–6679, 2020.

[18] A. Ahmad, M. Saad, L. Njilla, C. Kamhoua, M. Bassiouni, and A. Mohaisen, "BlockTrail: A Scalable Multichain Solution for Blockchain-Based Audit Trails," in *Proceedings of the International Conference on Communications (ICC)*, 2019, pp. 1–6.

[19] M. Gamal, R. Rizk, H. Mahdi, and B. E. Elnaghi, "Osmotic Bio-Inspired Load Balancing Algorithm in Cloud Computing," *IEEE Access*, vol. 7, pp. 42 735–42 744, 2019.

[20] J. van de Belt, H. Ahmadi, and L. E. Doyle, "Defining and Surveying Wireless Link Virtualization and Wireless Network Virtualization," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1603–1627, 2017.

[21] K. Ogawa, K. Kanai, K. Nakamura, H. Kanemitsu, J. Katto, and H. Nakazato, "IoT Device Virtualization for Efficient Resource Utilization in Smart City IoT Platform," in *Proceedings of the International Conference on Pervasive Computing and Communications Workshops*, 2019, pp. 419–422.

[22] Y. Cheng, C. Zhu, and F. Peng, "Wireless Virtualization for Energy Harvesting Aided Internet of Thing with Multi-Operator," in *Proceedings of the Information Communication Technologies Conference (ICTC)*, 2020, pp. 44–48.

[23] T. Bahreini, H. Badri, and D. Grosu, "Mechanisms for Resource Allocation and Pricing in Mobile Edge Computing Systems," *IEEE Transactions on Parallel and Distributed Systems*, pp. 1–1, 2021.

[24] H. Wang, Y. Ma, X. Zheng, X. Chen, and L. Guo, "Self-Adaptive Resource Management Framework for Software Services in Cloud," in *Proceedings of the Intl Conf on Parallel Distributed Processing with Applications, Big Data Cloud Computing, Sustainable Computing Communications, Social Computing Networking (ISPA/BDCloud/SocialCom/SustainCom)*, 2019, pp. 1528–1529.

[25] S. K. Battula, M. M. O'Reilly, S. Garg, and J. Montgomery, "A Generic Stochastic Model for Resource Availability in Fog Computing Environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 4, pp. 960–974, 2021.

[26] M. M. Than and T. Thein, "Energy-Saving Resource Allocation in Cloud Data Centers," in *Proceedings of the Conference on Computer Applications(ICCA)*, 2020, pp. 1–6.

[27] R. A. Alzahrani, S. J. Herko, and J. M. Easton, "Blockchain Application in Remote Condition Monitoring," in *2020 IEEE International Conference on Big Data (Big Data)*, 2020, pp. 2385–2394.

Riya Tapwal is a Ph.D. Research Scholar in the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, India. She has completed her M.Tech degree in Mobile Computing from the National Institute of Technology, Hamirpur, India, in 2020. Prior to that, she received the B.Tech degree in Computer Science and Engineering in 2018 from University Institute of Information Technology, Himachal Pradesh University, Shimla, India. The current research interests of Riya include Wireless Networks, Fog Computing, Industrial Internet of Things, and BC.

Pallav Kr. Deb is a Ph.D. Research Scholar in the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, India. He received his M.Tech degree in Information Technology from Tezpur University, India, in 2017. Prior to that, he has completed the B. Tech degree in Computer Science from the Gauhati University, India, in 2014. The current research interests of Mr. Deb include UAV swarms, THz Communications, Internet of Things, Cloud Computing, Fog Computing, and Wireless Body Area Networks. His detailed profile can be accessed at https://pallvdeb.github.io

Sudip Misra (SM'11) is a Professor at IIT Kharagpur. He received his Ph.D. degree from Carleton University, Ottawa, Canada. Prof. Misra is the author of over 350 scholarly research papers. He has won several national and international awards, including the IEEE ComSoc Asia Pacific Young Researcher Award during IEEE GLOBECOM 2012, the INSA NASI Fellow Award, the Young Scientist Award (National Academy of Sciences, India), Young Systems Scientist Award (Systems Society of India), and Young Engineers Award (Institution of Engineers, India). He has also been serving as the Associate Editor of the IEEE TRANSACTIONS ON MOBILE COMPUTING, the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, the IEEE TRANSACTIONS ON SUSTAINABLE COMPUTING, the IEEE SYSTEMS JOURNAL, and the INTERNATIONAL JOURNAL OF COMMUNICATION SYSTEMS. Dr. Misra has 11 books published by Springer, Wiley, and World Scientific. For more details, please visit http://cse.iitkgp.ac.in/∼smisra.

Surjya Kanta Pal completed his graduation from Government Engineering College, Jalpaiguri, West Bengal in 1991, M.Tech from IIT Kanpur in 1993, and Ph.D. from IIT Kharagpur in 1999, in Mechanical Engineering. Professor Pal has done 3 years of Postdoctoral research at The University of Sheffield, UK. His current research interests include Friction Stir Welding, Industry 4.0, Modelling, and simulation of manufacturing processes. He started teaching at IIT Guwahati from July 2002 to July 2004 and then joined IIT Kharagpur, where he is presently working as a Professor in the Department of Mechanical Engineering. Professor Pal has published 264 research articles which include 156 International Journal Papers, 15 International Book Chapters, and 90 Conference articles. He has also filed 11 patents, out of which five are in the area of Industry 4.0.