PhobosBC: A Blockchain-based Crowdsourced Post-disaster Mapping System and its Agent-based Simulation

Raunak Sarbajna
rsarbajna@uh.edu
University of Houston
USA

Christoph F. Eick
CEick@uh.edu
University of Houston
USA

Aron Laszka
aql5923@psu.edu
Pennsylvania State University
USA

ABSTRACT

After a major natural disaster strikes a region, emergency response often lacks information about the post-disaster state of the road network. Conflicting information about the region from multiple sources creates confusion. Thus, it is imperative to obtain an updated map, which provides accurate information about currently navigable paths and identifies hazardous locations. However, due to possible damage to communication and computation infrastructure, existing centralized geospatial services may provide outdated information. One possible solution to this problem is using Internet-of-Things based devices that are deployable without relying on a centralized service, which will be able to both acquire and disseminate local data. In this paper, we introduce PhobosBC, a decentralized system which is able to provide a reliable mapping service using volunteer work, while relying on a blockchain backend. This solution utilizes the availability of modern smartphones with GPS receivers and processing capabilities to collect sequences of GPS locations and combine them into trajectories. These trajectory data are submitted as entries into a blockchain after processing them through a smart contract. PhobosBC relies on the inherent robustness and distributed nature of blockchains to make collating and assembling a map from these paths more accurate and less susceptible to disruption. Compared to a centralized system, PhobosBC is more resilient and is able to respond to individual agents going offline while still retaining consensus. As a performance incentive, we rank volunteers by the amount of correct data submitted and publicly recognize top performers. We develop an agent-based model to simulate PhobosBC under a hypothetical disaster scenario, and we compare our approach with a crowdsourcing system which deploys a simulated central control mechanism. We share the problems faced during creation of the agent-based model and the lessons we learned. Our results show that PhobosBC is able to recreate the ground truth faster while being more fault tolerant.

CCS CONCEPTS

• **Information systems → Geographic information systems**.

KEYWORDS

Agent Based Models for Spatial Simulation, Spatial Analysis based on Simulation, Blockchain, crowdsourcing, Internet of Things, disaster response, disaster map building

GeoSim '23, November 13, 2023, Hamburg, Germany

ACM Reference Format:
Raunak Sarbajna, Christoph F. Eick, and Aron Laszka. 2023. PhobosBC: A Blockchain-based Crowdsourced Post-disaster Mapping System and its Agent-based Simulation. In *6th ACM SIGSPATIAL International Workshop on GeoSpatial Simulation (GeoSim 2023) (GeoSim '23), November 13, 2023, Hamburg, Germany*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3615891.3628016

1 INTRODUCTION

As human population increases, the number of communities affected by major natural disasters rises. Despite advances in predictive models for adverse weather, much effort has to be devoted to both mitigation and reactive systems by emergency management agencies [1]. After a disaster strikes, preliminary information emerging from the affected area can be inaccurate and often has inconsistent data, which need to be processed urgently to make it usable. As a disaster affected area has undergone significant changes, the most immediate need is to quickly create a first-order approximation of the ground details [2]. Emergency response and government officials need this critical information so that they can divert aid to regions that are in most need of support. A disaster-affected region is not always conducive to thorough surveys by external observers to create such a report [3]. Thus, the best and most logical source of information is the affected people themselves and any civilians already in the area.

In order to obtain data from civilians, we rely on crowdsourcing [4]. Generally, most people have access to smartphones, which have a multitude of useful sensors such as GPS receivers, accelerometers, gyroscopes, ambient light sensors, and magnetometers. Collecting sensory data can give crucial information about the state of the immediate location. Most modern IoT devices rely on similar sensor hardware but are implemented on a global network of smart devices. They generate massive amounts of data, which are utilized by various crowdsourcing platforms [5]. Erroneous data, caused by malfunctioning sensors or untrustworthy actors, will contribute to lowering the accuracy of the information. As the size of the data increases, maintaining integrity will become a challenge.

The question arises about what incentives people should receive to volunteer their time and resources to such a project. We find that there are many real-world examples of people being altruistic and travelling from great distances to offer aid [6]. An example of a volunteer-based, successful post-disaster crowdsourcing system is CrowdSource Rescue, a Houston-based non-profit which offered GPS-based tracking and mapping services to connect civilian rescue workers with people in urgent need during Hurricane Harvey [7]. The system performed rapid relief and recovery operations and helped lessen the load on governmental emergency services. Still,

a problem emerges with the collating of multiple types of data and keeping them organized and referenced.

In this paper, we propose PhobosBC, a system that can quickly and efficiently create navigable maps of an area utilizing contributions from volunteers. PhobosBC helps rescue teams map a complex terrain with shifting conditions, with active knowledge about available paths. It employs a novel blockchain based architecture that works with greater efficiency than existing, traditional edge-computing based systems. It features decentralization as a core component. It has greater robustness, fault tolerance, and resilience with respect to traditional crowdsourcing difficulties in comparison with a traditional centralized, single point-of-failure systems. As an incentive mechanism, we rank the quality of contributions of individual volunteers and publicly acknowledge top performers.

We evaluate PhobosBC's performance using an agent-based simulation system. The existence of detailed post-disaster maps, such as U-Flood [8] enables us to realistically perform such a simulation. We use these maps as ground truth for our simulation. We present experimental results which demonstrate that PhobosBC produces maps more quickly and reliably compared to a central-authority based system.

The rest of this paper is structured as follows. Section 2 introduces the crucial requirements and assumptions that we must take into consideration when developing our solution. Section 3 provides an overview of existing mobile crowdsourcing schemes, both blockchain and non-blockchain based. We then describe the system architecture and detailed protocol of PhobosBC in Section 4. In Section 5, we use an agent-based model to simulate PhobosBC under a hypothetical disaster scenario; moreover, we also compare our approach with a crowd-sourcing system that relies on a central control mechanism. We also talk about the challenges we faced during model construction and deployment. We conclude and discuss avenues for further research in Section 6.

2 CROWDSENSING REQUIREMENTS

A post-disaster response system will naturally need to deal with some common situations. Regular people, registered as data collectors within PhobosBC, will be the primary agents of the data collection process. To complete the task, they will need to physically travel to all accessible locations, automatically collect or manually enter the data and add it to the blockchain. As such, PhobosBC is designed to work under the conditions which will be described in the remainder of this section.

2.1 Assumptions

Disruption to communication networks. Any major natural disaster will damage a significant portion of telecommunication infrastructure [9] within the affected area and thus render existing communication methods unreliable [10]. Additionally, there can be severe impacts to underground cabling, overground towers and street poles [11]. We assume that some of the existing network towers are no longer capable of accurately and precisely delivering messages.

Disruptions to infrastructure. As there will almost certainly be widespread power outages [10], it is possible that street lights, deicers, or traffic lights will go down for an indeterminate period of time. This is why automated collection of ground information that digital apps rely on are disrupted in the immediate aftermath of the disaster, just when people are most vulnerable. There might be catastrophic damage to many buildings in the area, with especially stronger impacts to those in the direct path of the disaster. Lack of power will create a scarcity of drinking water, which may take months to restore. Damaged natural gas pipelines can make entire areas become uninhabitable for weeks.

Availability of communication devices. We can safely assume that every volunteer in the field will have access to some sort of computation device that is capable of wireless communication such as, but not limited to, a cell phone. The device would be capable of having short-range, ad-hoc wireless communication with similar nearby devices without relying on an internet connection.

Availability of volunteers. We must assume that there will be large groups of volunteers, both from within and outside the affected area, who will rush to provide aid. These are mostly non-governmental, loosely organized peoples who will be coming to offer whatever help they can. Studies show that "voluntary mutual help motivated by altruism" [6] is a strong force. Any system that seeks to provide aid should leverage these people.

2.2 Objectives

If the previously stated assumptions hold, then a system designed to work during a natural disaster must realize the following objectives.

Robustness. Reliability is the most important characteristic of the system. What result the system generates, whether it be a navigation map, population density estimates, etc., must be completely trustworthy [12]. Additionally, incorrect results must not invalidate the final output and the system must be able to maintain limited functionality in case of major systemic failures. It must account for the fact that real-world data collection is messy and error-prone. It should be able to appropriately handle missing data and dismiss any corrupted data. In case of communication losses, the system must be designed to fail safely.

Support for collaborative problem solving. Society after a natural disaster is chaotic. Any system designed to work under such an environment needs to be able to take advantage of volunteers pooling their resources. Distributed problem solving using both human and technical resources is essential to collect up-to-date information and provide them to emergency response so that they can take the most appropriate action [13]. This might lead to scalability problems, so the system must be also able to scale up quickly to deal an evolving situation involving a large number of volunteers [14].

Decentralization. In order to achieve the first two objectives, the system must be decentralized. We assume that any established authority might not be available once the disaster is under way [15]. We are not guaranteed to have access to one that has existed since before the disaster. Requiring a central authority to be set up during the disaster is also not viable [16]. So, the system must work without external verification. This will also have the advantage of any avoiding complications arising from an adverse third party. The system must be able to deal with attackers both benign and malicious.

Table 1: Comparison of Existing Solutions with our Requirements

	Requirements	Existing Approaches							
		Yuan et al. [18]	Ming et al. [19]	Yang et al. [20]	Kucuk et al. [4]	Bhattacharjee et al. [21]	Sarbajna et al. [22]	Frigerio et al. [1]	Tan et al. [23]
Assumptions	Disruption to Communication Networks				✓	✓	✓	✓	
	Obstructions to Area Infrastructure				✓	✓	✓	✓	
	Availability of Communication Devices		✓		✓	✓	✓		✓
Objectives	Robustness	✓	✓	✓			✓		
	Collaborative Problem Solving	✓	✓	✓	✓	✓	✓		✓
	Decentralization	✓	✓	✓			✓		✓
	Energy efficiency	✓	✓			✓			
	Transparency			✓					✓

Energy efficiency. The method must be energy and computationally efficient, and have a hard limit on its allowed energy consumption. Any disaster scenario will necessarily be a place of energy shortages, and the system must work on top of existing challenges and not consume more than the value it provides. Wireless transmissions methods' inherently energy inefficiency must be minimised by not relying on broadband broadcasts and instead relying on point to point communication methods. Low energy consuming technologies, such as Bluetooth communication or Near field communication must be preferentially used. If peer-to-peer communication is used, the absolute minimum number of messages required to complete the result must be transmitted [17]. The system must be optimized to work on low-power modes and use energy efficient algorithms for its calculations. The system must recognize when it is exceeding its operational energy computation limit and stop.

Transparency. All the information collected must be made available, upon request from higher authorities, for scrutiny and audit after the disaster situation has stabilized enough for those higher authorities to establish themselves. It is necessary that the decision making process is fully recorded so that it is possible to later reconstruct exactly how and what information the decision was derived from.

3 EXISTING SOLUTIONS

We conducted a review of the related literature and evaluated which approaches effectively meet our assumptions and objectives. The results can be found in Table 1. We see that non-blockchain based solutions, which use Internet-of-Things based approaches in an Edge computing-like network, tend to fulfil our initial assumptions, but fail with many of our critical requirements. Complete decentralization, which is very important in a confused post-disaster scenario, is not assumed for neither of Kucuk et al. [4] or Bhattacharjee et al. [21]. Both require multiple local authorities and at least one central authority.

Blockchain based solutions like the one proposed by Yuan et al. [18], Ming et al. [19] and Yang et al. [20] however, assume that their operating environment is very much a modern technological city, with all its attendant infrastructure. They would all fail if we assume the networks and power have become unavailable. They are, on the other hand, designed to not rely on any central authority of any kind and have transparency built-in. Zebralancer [18] relies on randomness for its cryptographic technique but it ensures that the results are unchangeable once encrypted. It can thus guarantee determinism within the protocol. Yang et al., however randomly assigns workers to tasks for anonymity and thus it cannot guarantee the same.

One major issue is that crowdsensing methods are less strict on energy efficiency. They mostly use very computationally expensive techniques for their ledger and encryption methods, which is understandable given their usual operating environment. Their communication methods broadcast usually over wifi or piggyback over existing cellular connections, both of which are not very power efficient. Making effective use of point to point communication and using low energy techniques like Bluetooth 5 is not widespread.

Sarbajna et al. [22] have proposed DeimosBC, which is a crowdsourcing system that can build a map from volunteer submitted GPS points and utilize the decentralized nature of blockchains to improve upon standard practices. It relies on its smart contract to generate paths from noisy GPS points and combine them into a navigable map.

Zebralancer [18] is a private and anonymous decentralized crowdsourcing system which tries to solve two challenges of decentralizing crowdsourcing: data leakage and identity breach. They designed a methodology called outsource-then-prove which attempts to resolve the conflict between the blockchain transparency and the data confidentiality. Their approach satisfies the basic utilities/fairness requirements of data crowdsourcing, which ensures that any requester will not pay more than what they deserve and that each worker gets a payment if they submit data to the blockchain.

CrowdBC [19] is a blockchain-based decentralized framework for crowdsourcing, in which a requester's task is solved by a crowd of workers without relying on any third-party trusted authority. It guarantees users' privacy while ensuring low transaction costs. The system prototype was deployed on a Ethereum public test network and tested using the CIFAR-10 dataset.

Mobile Applications for Emergency Response and Support (MAppERS) [1] is an Android application that enables on-site storage capacity of real-time data and a platform to support rescue operations during a crisis. Citizens are able to provide the location and status of the people currently affected by the flood and then upload photos or provide local water-level assessments. This information is critical for emergency responders to organize and prioritize interventions. MAppERS provides different modules for volunteers and for citizens who are the first actors on scene in any surveillance strategy. The system is used to train the exposed population in flood awareness and help understand the proper terminology to explain what is happening. It aims to promote awareness within the local population of distributed, crowdsources surveys. The app is also designed to be used in time of crisis to provide communication between the civilians and first responders.

Bhattacharjee et al. [21] propose a complete Post-Disaster Map Builder, which uses a crowdsensed digital pedestrian map construction system using a smartphone based DTN. They propose

collecting trajectory traces through volunteers, which are then periodically shared among the other volunteers within the disaster affected area through a custom network solution. They gradually construct pedestrian maps of the affected areas by combining these traces over time. The results show that their system is capable of constructing digital pedestrian maps of a disaster affected areas with high accuracy at the cost of marginal delays.

Tan et al. [23] propose a blockchain based trusted payment service mechanism for a crowdsensing system, which they claim is completely decentralized. Yang et al. [20] propose a novel solution, dubbed a blockchain privacy-preservation crowdsensing system, to address these privacy problems in traditional crowdsensing systems. Kucuk et al. [4] have designed a post-disaster framework using the IoT communication technologies for disaster management based on a crowdsensing clustering algorithm.

For post-disaster agent-based modelling specifically, we looked at some practical examples. Koch et al. [24] used data to construct a microsimulation model of ambulance response in New Windsor, NY. As Emergency Medical Services (EMS) rely on road networks to respond to patients quickly, they developed a model to help EMS improve disaster preparedness. The model showed that removing any of three sets of links (the set of four bridges in New Windsor, geographically central roads, and the most frequently traveled roads) increased response time, but removing the most frequently traveled links had the greatest effect. We looked at the survey of disaster response models by Bae et al. [25], where they discuss how an agent-based model can be used to evaluate the efficiency of disaster responses to MCIs. Their proposed model includes geospatial details and medical information and the case study in the paper shows how the model can be used to analyze a disaster response system. The proposed method can provide insights into disaster response systems and can be used to improve them.

4 PHOBOSBC: OUR PROPOSED PROTOCOL

Based on our assumptions, we build a crowdsourcing system to create a post disaster map. We detail the architecture and workflow of our system next.

4.1 System Architecture

We propose PhobosBC, a system where certified volunteers combine their knowledge to automatically generate a working digital map of accessible areas within the disaster affected region. The objective of PhobosBC is to ultimately generate a complete and correct map of accessible paths through a disaster affected region through data collected by volunteers equipped with an IoT enabled device. In our work, a map is defined as an undirected graph with vertices representing locations which are described by longitude-latitude pairs and edges being traversable paths between those locations. The blockchain itself would at any time contain the current partially-complete graph. *Data collectors* submits a trajectory and *data processors* run a smart contract to check whether the submitted route is valid and adds it to the chain if so. *Trajectories* are sequences of GPS locations collected by a volunteer. Each consecutive trajectory point is separated by a threshold distance. The threshold distance is the minimum distance between collection points, required for segmenting the paths. Lower values create more fine grained maps. A combination of all these partial maps create a network of interconnected paths that will allow people to navigate a difficult terrain in the real world. The final, complete map may be subtly different from the pre-disaster map as it will not contain those paths that have been blocked due to some disaster.

At any point of time, the blockchain can contain multiple disconnected subgraphs, which represent partial maps. These maps are represented as graphs with each vertex containing: (i) which vertices it is connected with, (ii) real-world location, and (iii) additional sensor data. These are created by individual Data Collectors submitting their data. Once all the processors are satisfied that no more updates can be done, we freeze the blockchain [26] and make the map available for general public use. This process can repeat as many times as needed, generating up-to-date maps while there are still volunteers. The system is halted when the Data Processors detect that the system has too many erroneous data being submitted and can compromise the integrity of the map.

In order to provide a lightly gamified incentive mechanism, we make available the number of data points submitted by each volunteer, and their accuracy as determined by the rate of erroneous trajectories. We periodically publish an updated ranked list of the top volunteers.

Next, we define the parties involved in PhobosBC and their structure.

4.1.1 Data Collector. Any volunteer coming into the system can perform as Data Collectors. They do not need any verification from an outside authority, such as the local Government, but they must be registered into the blockchain system heading out for work. As they are registered, they are each randomly assigned a unique identity. They will then start roaming the region and collect data points which will later be collated into traces. Collection of consecutive trajectory point is separated by a threshold distance, which is managed by the *Trace management system* as described in Section 4.3.2. After collection, that trace is then broadcast to all local nodes that are within range. Until it receives confirmation of delivery and successful processing from at least one Processor, all subsequent traces are held in a queue. The Data Collector keeps working until they voluntarily stop or receive a notification from the Trace Management of errors in the data collected. If the number of errors exceeds a preset limit, the user must be re-authenticated into the system.

4.1.2 Data Processor. Data Processor are assumed to have greater data processing capabilities and thus several responsibilities:

(1) *Package Verification*: Accepts only data-packets from the Collectors that have been correctly formatted. Performs error checking.
(2) *Author Identification*: Performs verification of incoming data packets and acknowledges receipt.
(3) *Trace Processing*: Operates the Trace Management System (described in Section 4.3.2), and passes it on for further processing to the Data Summarization System (described in Section 4.3.1). Runs the smart contract.
(4) *Result Provision*: Provides generated map to any requester.

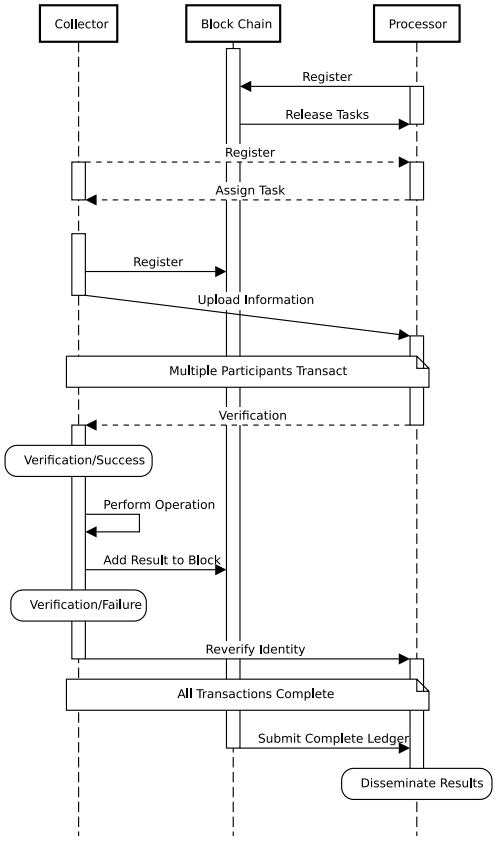

Figure 1: Flow of control in PhobosBC.

4.2 System Security

We assume that PhobosBC is not a target for malicious actors. We provide no service that can be compromised to adverse ends or monetary gain. People are inherently altruistic and come together during as a community to support each other during natural disasters [6]. The inherent protections from the distributed ledger suffice for our purposes. However, a major secondary issue is the integrity and correctness of the data itself. Volunteers may submit incorrect data due to multiple reasons other than self-interest, and PhobosBC is designed to mitigate this. Even assuming that most of the submitted data is correct, we need to deal with outliers. Our smart contract performs a point verification process to deal with this.

4.3 Protocol

PhobosBC is an extension of the Bhattacharjee et al.'s centralized post-disaster map-builder system [21]. Construction of a comprehensive digital pedestrian map of an entire region first requires building local pedestrian maps of the disaster affected areas and then inferring the global pedestrian map by collating the local maps. The components of PhobosBC are described in the subsections below.

4.3.1 Data Summarization System. Each Data Processor is responsible for collating the data that it receives in the form of traces from Data Collectors. This module is then used to construct the disaster map from those traces. It checks for overlaps and excessive gaps between points. Traces are then checked for consistency and erroneous traces are discarded. It then combines the remaining traces into a single map. For the purposes of security, it also keeps a tally of the number of the traces submitted per data collector. This process is designed to be lightweight and efficient.

4.3.2 Trace Management System. Each Data Collector periodically collects their own GPS location. These collected data create sequences of trajectory points, which are combined into traces based on temporal and physical distance between them. To avoid oversampling, we sparsely sample the data. This process is susceptible to random noise because the Data Collectors do not always move at a uniform speed. During the course of their journey they tend to stop occasionally. We pre-define a minimum displacement between each collected trajectory point, depending upon the nature of the region. This has two separate purposes: (i) Reduces the possibly of random errors introduced into the collection due to arbitrary halts (ii) As GPS receivers have very high energy consumption, reduced and periodic polling saves power.

We also use the concept of *trajectory segmentation* [27]. Sometimes displacement between two consecutive trajectory points becomes much greater than the minimum displacement due to the missing/bad GPS sensor data. This sporadic gap, which might affect the direction of the trace between the consecutive trajectory points produces erroneous trajectory traces. In order to avoid such gaps in trajectory traces, we segment trajectory up to that gap and generate smaller trajectory traces. We use a threshold distance, that is larger than the minimum displacement, between consecutive trajectory points for trajectory segmentation [28].

4.3.3 Blockchain Manager. The Blockchain Manager is responsible for handling the running of the blockchain protocol, and is present within the *Data Processors*. All Data Collectors who will be going out into the field must be pre-verified, without compromising their identity or their exact location. Drawing upon the central tenet of a blockchain protocol, we do not have any central authority who decides consensus or resource allocation. It is the responsibility of the Data Processor to run the smart contract and submit the current partial map to merge with all other geographically distinct complete maps. The smart contract is also responsible for running a route tester to ensure every single location is visitable.

4.4 Workflow

In this section, we present our blockchain-based extension to the Bhattacharjee et al.'s map builder system. The flow of control of the proposed system has been depicted in Fig. 1 which calls the following operations:

- *Registration*: All collectors and processors will need to register with the blockchain system at initialization. Each of these registered collectors will be assigned a pair of cryptographic keys, and their identities are noted. The data processor is allotted a special token for identification purposed. All the registration information is recorded as a transaction in the blocks.

- *Task Assignment*: This is an automated process performed by the *requester*. Once registration is complete, all collectors receive instructions about what data they are collecting and a randomized starting point based on their geographic location. This might include just GPS points, or have additional sensor data. They are then free to start doing their tasks. The task includes the starting trajectory point (which may be changed as per their need), the direction of the trace, the current time and the status.
- *Smart Contract Creation*: The data processor is responsible for running the smart contract which checks quality of the sensory data coming in from Data Collectors, in accordance with the Data Summarization system Section 4.3.1.
- *Uploading Sensory Data*: Data Processors upload the received sensory data to the blockchain.
- *Loading Tasks*: The processor downloads all the information from the latest map and posts all the information on their private blockchain network. The processor is responsible for maintaining the consistency of the task information on the blockchain (the complete global map and their own partially-complete local map).
- *Broadcast*: If the current state of the generate map stored in the blockchain is satisfactory, then the map is made available for download from any requester by a *server* node. The tally of data points submitted to the chain by each Data collector is updated.
- *Retirement*: The data processor constantly checks the failure rates of all other nodes. To avoid degradation of output quality, once the number of failure exceeds a threshold, the blockchain is frozen. The final block is marked to ensure all subsequent blocks get automatically discarded. This freeze state is broadcast throughout the network from an authorized Processor and is amplified by every node who received it, whether Processor or Collector. Other Processors may run their own integrity tests to ensure the decision is correct. A majority vote is then undertaken to finalize the decision to freeze or not.

5 PROTOTYPE IMPLEMENTATION AND EVALUATION

In order to assess how PhobosBC would perform in a real-world setting, we design an agent based model to simulate a specific disaster scenario. For the purposes of the simulation, each volunteer will be an agent in the system. We implement *Data Processors* and *Data Collectors* as agents exploring a graph with the same layout as the roads of a city. These agents have the capacity to communicate with each other and run the smart contract. We build two different simulation models, one for PhobosBC and one which assumes centralized control.

The controllable parameters of the simulation are the number of agents m, the total number of time-steps per run t, the failure rate f (probability that an agent goes down for 3 consecutive time-steps) and τ (the number of points collected by a data collector in a single simulation step). A single simulation step represents the movement of an agent from one node to the next.

We simulate the performance of PhobosBC to determine the effectiveness and accuracy compared to an idealized centralized system. The experiments were performed on a local machine running on Windows 10 with an Intel i9-9880H processor, 32 GB of DDR4 RAM and an NVMe drive with 8.0 GT/s. The simulation was performed using the Agent Based Modelling library MESA, written in Python [29]. The smart contracts were written in Solidity [30] and tested using Ganache running the Ethereum blockchain [31].

For ground truth, we utilized the Harvey Damage Assessment GIS dataset from the City of Houston [32]. For our base layer we used a city road network map from OpenStreetMaps, through OSMnx [33]. Using flood damage reports we generate non-passable roads in those road networks. The city model is a road network graph imported from Open Street Maps, simulating a section of Houston. The graph is stored as NetworkX objects [34], where roads are edges and intersections are vertices. The vertices stores information: a reference ID, location information (in latitude/longitude) and height of maximum water inundation level. Our generated map is then compared against this flooded city road network map using the evaluation procedure described later in Section 5.3.

5.1 Error Model

The simulation model for PhobosBC requires Data Collectors to acquire a group of synthetic GPS points. GPS data is inherently noisy, and contain uncertainties in their calculated position, depending on factors such as latitude, seasons and continental water loading. For accurate simulation, that noise must be added to the synthetic data. GPS noise models have been widely studied and are composed of several types of noise: white noise, flicker noise, random-walk noise. White noise dominates at all frequencies over the others, so that is what we use.

We model the error by assuming that the raw GPS point is G_0 with the additive noise in the form:

$$G_t = G_0 + N_t \tag{1}$$

where N_t is an independent stochastic process which is normally distributed with zero mean and variance of σ^2. We obtain realistic values of σ^2 from the Central Error values found in Table IV in Lee et al. [35]. We pick the values for different consumer devices and and assign them to a Data Collector. In order to better simulate the random nature of crowdsensing, we also add a 10% chance of no location being collected at all.

5.2 Point Verification Method

In order to de-noise and remove errors from the collected GPS data, we leverage the capabilities of a crowdsourced system. We also assume that we have access to a pre-disaster map M, and use the information from that as control points to anchor our collected data. Our aim is to generate a set of *verified* points.

For every single point collected by a Data Collector, we aim to locate the closest point in the original pre-disaster map that is within a distance threshold; if no such point exists the point will be discarded. The threshold distance is the maximum tolerated GPS noise. Moreover, only points that have been collected by at least three different data collectors will be considered verified. See Algorithm 1 for details.

Algorithm 1 Point Verification Pseudocode

Input: Set of Data Collectors D
Input: Points P_d collected by Data Collector d
Input: Pre-disaster Map M
Input: Threshold distance θ
Output: Bag of Verified points $\mathcal{V}$

```
for all Data Collector d ∈ 𝒟 do
    for all Points p ∈ P_d do
        Compute closest point r ∈ M to p
        if d(r, p) ≤ θ then
            Add r to 𝒱
        end if
    end for
end for
for all points r ∈ 𝒱 do
    if r has not been collected by three different Data Collectors then
        Remove r from 𝒱
    end if
end for
```

5.3 Evaluation Metric

To evaluate the accuracy of the map generated by PhobosBC, we will compare it against a post-disaster map from a known source. However, paths constructed by PhobosBC and paths in the ground truth maps may not necessarily agree in their edges. Consequently, rather than comparing edges, we check whether each path in the original map has a corresponding connected sequence of edges in the reconstructed map. The percentage of paths in the ground truth map which were found in the reconstructed map is used as the evaluation metric.

5.4 Results

We initialize PhobosBC by distributing m agents, of which 85% are Data Collectors and 15% are Data Processors, randomly over the map. We run different experiments by varying m between 50 to 1000, and t between 200 to 600. When the simulation starts, all the Data Collectors start recording their current location and make a connection to the closest Data Processor. They are permitted to move to one adjacent edge during one time step. r is fixed to 3 during the experiments. All Collectors send the last 3 collected points to the closest Data Processor. Transmitting this data takes one time step, which can be done while the Collector is starting another sequence. There is no queuing mechanism: all collected data are transmitted at once. The Data Processor runs the smart contract and submits the collected data to the chain, after performing point verification as described in Algorithm 1. For most experiments, we set the failure rate f to 2%.

For comparison, we generate an idealized centralized model. In this case, there is just one Data Processor, and that agent is available to receive data from every Data Collector on the field. Similar to the standard model, the central data processor also has a 2% chance of going down. In order to simulate the irregular availability of a central point, we give it a 25% chance of coming back after every 3 time-steps. We evaluate this setup against our standard

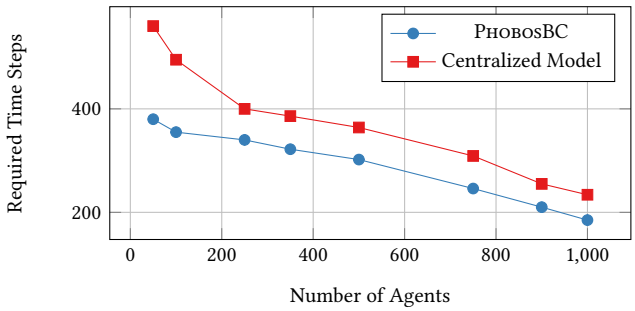

Figure 2: Time steps required to reach 95% reconstruction of the original map.

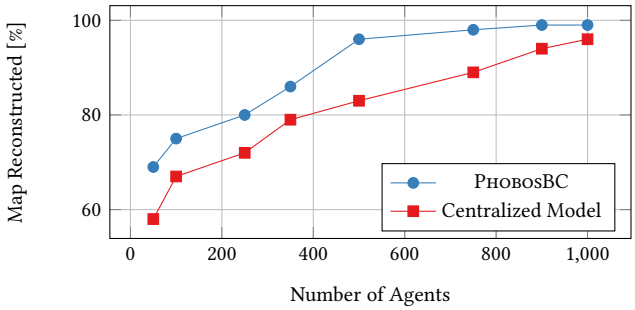

Figure 3: Map reconstruction percentage of PhobosBC vs. Centralized Model at 350 time steps at various number of agents.

setup and compare how long it takes to to reconstruct 95% of the ground truth map. The results can be seen in Fig. 2. We can see that PhobosBC performs consistently better than the centralized model over various agent numbers. We can also see that PhobosBC is also capable of reaching high accuracy levels relatively quickly using very few agents.

For our baseline comparison, we limit $t = 350$ for both models and then vary the number of agents m between $[50, 1000]$. The results of this experiment are shown in Fig. 3. It is clear that although both models have poor accuracy with low agent numbers initially, PhobosBC is quickly able to reconstruct a large portion of the map. Using PhobosBC, roughly 500 agents are capable of accurately recreating the ground truth in 350 time steps, making it faster than the centralized approach by an average of 50 time steps.

The next experiment tests the robustness of the two systems by varying the failure rate f. Moreover, we analyze how many volunteers are necessary to avoid mapping performance degradation. We do this by varying the node failure rate f between 2% and 50%, while keeping the length of the simulation fixed at $t = 350$ timesteps and also vary the number of total nodes as follows: $m = 100, 250, 500, 1000$. In our analysis, we assume that the map reconstruction is unsuccessful if the reconstruction rate falls below 75%. The results of this experiment are depicted in Fig. 4. We observe that PhobosBC is able to maintain its performance far longer than the centralized approach. We can see that the system is able to tolerate up to 35% to 40% node failure. However, when approaching

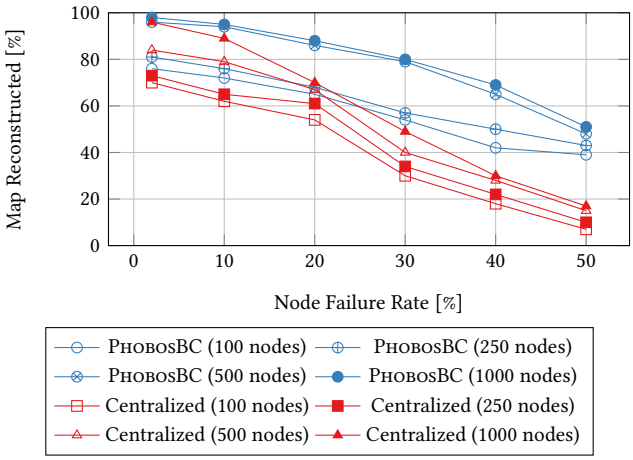

Figure 4: Map reconstruction percentage of PhobosBC vs. Centralized Model at 350 time steps with increasing failure rates.

50% failure rate the system is unable to maintain a high reconstruction rate. The centralized model performs far worse, as the central node going down irrevocably handicaps the entire system. Even a 20% node failure rate decreases its map reconstruction rate to below 60%. To test an extreme case, we run a single simulation with $m = 1000, t = 1000, f = 50$ for both models. The reconstruction rate for PhobosBC stabilizes at 59% while the centralized model stabilizes at 20%. This is slightly higher than the performance at $f = 50$ and $t = 350$. This indicates that simply letting the system run for long periods of time does not lead to a better map for high failure rates.

5.5 Challenges Faced

We found, during deployment and evaluation of our model, that agent-based modelling of city streets introduce very specific difficulties. The primary challenges we faced are *verifiability* and *reproducibility*. It can be difficult to verify that an agent-based simulation is accurate, as it is often difficult to know the true behavior of the real-world system that is being modeled. In order for the simulation to have real-world applications it is important to be able to reproduce the results. This can be difficult, as small changes in the parameters of the simulation can lead to significant changes in the results.

Our model also had to deal with *heterogeneity*, *non-linearity* and *uncertainty*. The agents in a city street model should be very heterogeneous, with different characteristics and behaviors, to be more realistic. This made it difficult to develop a single model that can accurately capture the behavior of all agents. The interactions between Data Processor/Collector agents within our city street model turned out to be non-linear, meaning that small changes in the behavior of one agent had large effects on the behavior of other agents. This made it difficult to predict the behavior of the system. Additionally there is often a lot of uncertainty about the parameters of a city street model, such as the speed of vehicles and the walking speed of pedestrians. This uncertainty can make it difficult to obtain accurate results from the simulation.

6 CONCLUSION

In this paper, we introduce PhobosBC, a novel post-disaster crowdsourcing system. Its system architecture relies on a blockchain to provide robustness and decentralization, and the ability for multiple disparate users to contribute their effort to a collaborative task. The system is designed to handle major disruptions to communication networks and obstructed infrastructure, requiring the end user to only have access to a simple mobile phone. The system has two types of internal users: Data Collectors and Data Processors, who perform the basic tasks of collecting information form the real world, processing it, and then adding it to the blockchain for anyone to access. This blockchain contains, at all times, the latest combined data product, and is made available to anyone who asks for it. Internally, the system relies on two modules: the Trace Management System, which processes the traces collected by the Collectors and the Data Summarization System, which collates the traces.

We use an agent-based modelling approach to simulate this system and compare it against an idealized centralized system working in the same space. Our experiments highlight the importance of setting macro-level agent behaviour over micro-level rules in order to more accurately create a baseline. The simulation results show a better performance and fault-tolerance compared to the centralized model. As far as future work is concerned, we are working on introducing crowd-movement simulations [36] within PhobosBC to more realistically model expected behaviour of our agents. We also plan to introduce more advanced blockchain capabilities into the system, such as complete confidentiality of users and resource management. We also aim to introduce an incentive mechanism within the system to reward volunteers for delivering more accurate maps more quickly.

REFERENCES

[1] S. Frigerio, L. Schenato, G. Bossi, M. Mantovani, G. Marcato, and A. Pasuto, "Hands-on experience of crowdsourcing for flood risks: An Android mobile application tested in Frederikssund, Denmark," *International Journal of Environmental Research and Public Health*, vol. 15, no. 9, p. 1926, Sep. 2018.
[2] J.-S. Huang and Y.-N. Lien, "Challenges of emergency communication network for disaster response," in *2012 IEEE International Conference on Communication Systems (ICCS)*, 2012, pp. 528–532.
[3] T. L. Henderson, M. Sirois, A. C.-C. Chen, C. Airriess, D. A. Swanson, and D. Banks, "After a disaster: Lessons in survey methodology from Hurricane Katrina," *Population Research and Policy Review*, vol. 28, no. 1, pp. 67–92, 2009.
[4] K. Kucuk, C. Bayilmis, A. F. Sonmez, and S. Kacar, "Crowd sensing aware disaster framework design with IoT technologies," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, no. 4, pp. 1709–1725, 2020.
[5] M. Durresi, A. Subashi, A. Durresi, L. Barolli, and K. Uchida, "Secure communication architecture for Internet of Things using smartphones and multi-access edge computing in environment monitoring," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 4, pp. 1631–1640, 2019.
[6] H. Kotani and M. Yokomatsu, "Natural disasters and dynamics of "a paradise built in hell:" a social network approach," *Natural Hazards*, vol. 84, no. 1, pp. 309–333, 2016.
[7] "CrowdSource Rescue." [Online]. Available: https://www.crowdsourcerescue.com
[8] "Interactive map shows where Harvey flooding is worst." [Online]. Available: https://www.cnet.com/news/maps-of-houstons-floods-shows-its-worse-than-you-thought-hurricane-harvey/
[9] World Economic Forum, "The global risks report 2020," World Economic Forum, Tech. Rep., January 2020.
[10] F. Kadri, B. Birregah, and E. Chatelet, "The impact of natural disasters on critical infrastructures: A domino effect-based study," *Journal of Homeland Security and*

Emergency Management, vol. 11, no. 2, pp. 217–241, 2014.
[11] A. M. Townsend and M. L. Moss, "Telecommunications infrastructure in disasters: Preparing cities for crisis communications," Center for Catastrophe Preparedness and Response, Robert F. Wagner Graduate School of Public Service, New York University, Tech. Rep., 2005.
[12] X. Zhang, Z. Yang, C. Wu, W. Sun, Y. Liu, and K. Xing, "Robust trajectory estimation for crowdsourcing-based mobile applications," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 7, pp. 1876–1885, 2013.
[13] H. Gao, G. Barbier, and R. Goolsby, "Harnessing the crowdsourcing power of social media for disaster relief," *IEEE Intelligent Systems*, vol. 26, no. 3, pp. 10–14, 2011.
[14] T. Kohler, "How to scale crowdsourcing platforms," *California Management Review*, vol. 60, no. 2, pp. 98–121, 2018.
[15] H. Khan, L. G. Vasilescu, A. Khan *et al.*, "Disaster management cycle-a theoretical approach," *Journal of Management and Marketing*, vol. 6, no. 1, pp. 43–50, 2008.
[16] F. Delkhosh, "A dynamic leader-follower model based on lack of central authority in emergency situations," *International Journal of Data and Network Science*, vol. 4, no. 1, pp. 73–90, 2020.
[17] C. Han and S. Armour, "Computational complexity and energy consumption analysis of dynamic resource scheduling algorithms for lte," in *75th IEEE Vehicular Technology Conference (VTC Spring)*, May 2012, pp. 1–5.
[18] Y. Lu, Q. Tang, and G. Wang, "Zebralancer: Private and anonymous crowdsourcing system atop open blockchain," in *38th IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2018, pp. 853–865.
[19] M. Li, J. Weng, A. Yang, W. Lu, Y. Zhang, L. Hou, J.-N. Liu, Y. Xiang, and R. H. Deng, "CrowdBC: A blockchain-based decentralized framework for crowdsourcing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 6, pp. 1251–1266, 2018.
[20] M. Yang, T. Zhu, K. Liang, W. Zhou, and R. H. Deng, "A blockchain-based location privacy-preserving crowdsensing system," *Future Generation Computer Systems*, vol. 94, pp. 408–418, 2019.
[21] S. Bhattacharjee, S. Roy, and S. D. Bit, "Post-disaster map builder: Crowdsensed digital pedestrian map construction of the disaster affected areas through smartphone based DTN," *Computer Communications*, vol. 134, pp. 96–113, 2019.
[22] R. Sarbajna, C. F. Eick, and A. Laszka, "DeimosBC: A blockchain-based system for crowdsensing after natural disasters," in *3rd Conference on Blockchain Research Applications for Innovative Networks and Services (BRAINS)*, 2021, pp. 17–20.
[23] L. Tan, H. Xiao, X. Shang, Y. Wang, F. Ding, and W. Li, "A blockchain-based trusted service mechanism for crowdsourcing system," in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, 2020, pp. 1–6.
[24] Z. Koch, M. Yuan, and E. Bristow, "Emergency response after disaster strikes: Agent-based simulation of ambulances in new windsor, ny," *Journal of Infrastructure Systems*, vol. 26, no. 3, p. 06020001, 2020.
[25] J. W. Bae, K. Shin, H.-R. Lee, H. J. Lee, T. Lee, C. H. Kim, W.-C. Cha, G. W. Kim, and I.-C. Moon, "Evaluation of disaster response system using agent-based model with geospatial and medical details," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 9, pp. 1454–1469, 2018.
[26] N. Shibata, "Proof-of-search: Combining blockchain consensus formation with solving optimization problems," *IEEE Access*, vol. 7, pp. 172 994–173 006, 2019.
[27] L. Zhu, J. R. Holden, and J. D. Gonder, "Trajectory segmentation map-matching approach for large-scale, high-resolution GPS data," *Transportation Research Record*, vol. 2645, no. 1, pp. 67–75, 2017.
[28] U. Blanke, R. Guldener, S. Feese, and G. Tröster, "Crowdsourced pedestrian map construction for short-term city-scale events," in *1st International Conference on IoT in Urban Space (URB-IOT '14)*, 2014, pp. 25–31.
[29] "Mesa: Agent-based modeling in Python 3+ – Mesa .1 documentation." [Online]. Available: https://mesa.readthedocs.io/en/stable/
[30] "Solidity — Solidity 0.8.10 documentation." [Online]. Available: https://docs.soliditylang.org/en/v0.8.10/
[31] "Ganache." [Online]. Available: https://trufflesuite.com/ganache
[32] M. F. Wehner, "Hurricane Harvey flood," Sep. 2019. [Online]. Available: https://www.osti.gov/biblio/1561271
[33] G. Boeing, "OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks," *Computers, Environment and Urban Systems*, vol. 65, pp. 126–139, Sep. 2017.
[34] "NetworkX — NetworkX documentation." [Online]. Available: https://networkx.org/
[35] L. Lee, M. Jones, G. S. Ridenour, S. J. Bennett, A. C. Majors, B. L. Melito, and M. J. Wilson, "Comparison of accuracy and precision of gps-enabled mobile devices," in *2016 IEEE International Conference on Computer and Information Technology (CIT)*, 2016, pp. 73–82.
[36] J. E. Almeida, R. J. F. Rossetti, and A. L. Coelho, "Crowd simulation modeling applied to emergency and evacuation simulations using multi-agent systems," *CoRR*, vol. abs/1303.4692, 2013. [Online]. Available: http://arxiv.org/abs/1303.4692