Leal *et al. Journal of the Brazilian Computer Society* (2021) 27:1
https://doi.org/10.1186/s13173-021-00104-y

Journal of the
Brazilian Computer Society

RESEARCH Open Access

A blockchain-based service for inviolable presence registration of mobile entities

Matheus Leal*† , Flávia Pisani† and Markus Endler†

*Correspondence:
mrol@aluno.puc-rio.br
†Matheus Leal, Flávia Pisani and Markus Endler contributed equally to this work.
Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), Rio de Janeiro, Brazil

Abstract

Several applications can benefit from recording information about the places a mobile entity visits and the length of time it spends there (e.g., shoppers, employees, buses, portable equipment, autonomous robots). This paper presents our approach to recording spatio-temporal presence information in a secure and inviolable way using a Distributed Ledger Technology. We implemented this solution as a middleware service that uses Complex Event Processing on smartphones to record beacon-smartphone proximity data in a blockchain efficiently. We have built upon the previous version of our service to include access control to the stored information. We analyzed the impact of this addition on the service's performance and observed that it introduced very little overhead while significantly increasing user privacy. Furthermore, we compared the effect of using different blockchain technologies on overall service performance and characterized scenarios where using either IoTeX or Ethereum can be suitable for this type of application.

Keywords: Blockchain, Presence information, Internet of Things, Mobile things, Middleware

Introduction

As a large portion of modern business and our social lives encompasses mobility, the places visited by people, goods, and vehicles become testimonials of our activities, goals, state of well-being, and even our intentions. The recent explosion of movable smart Internet of Things (IoT) devices was a key component to the formation of the Internet of Mobile Things (IoMT) [1, 2], turning the information about previous, current, and future positions of these devices into valuable knowledge in our modern economy.

Plenty of applications can benefit from recording spatio-temporal information about the current and previously visited places of mobile entities, be they employees, vehicles, portable machines, or autonomous robots. A few examples are task force and employee control, supply chain management, asset management, logistics, hospital operations, and public transportation.

In addition to that, to avoid manipulation, distortion, or fraud, some of these applications also require a permanent and inviolable record of places, when they were visited,

and how long each visit took. For instance, some companies may need to prove irrefutably that they serviced their clients at the correct place for a previously stipulated amount of time or that the maximum waiting time to be served has not exceeded the allowable limit. Similarly, a public transport service may have to prove that it has complied with its schedule, with each bus arriving and leaving each stop at the right moments. Furthermore, if a company sends a group of employees to a remote professional training session, it might want to check if some did not attend all the course modules.

Alternatively, consider the following service based on trusted mobility data that requires near to real-time automatic reaction. At universities, there is a deep concern about the privacy of employees. Therefore, the many security cameras installed around the departments' facilities should begin recording only when they detect suspicious movements among staff members. If an unauthorized staff member enters in a restricted access room (e.g., the graduate studies office or the director's office) and stays for more than the usual period, a Distributed Ledger should then register this person's presence in that room at that time as a definitive and inviolable record. Only after that should the corresponding cameras start recording in order to get other testimonial evidence. This online, reactive IoT application requires the immutable logging to be as fast as possible so that video cameras can start recording immediately after the movement and presence information is secured.

Another reason for immutable recording of location data to be fast (that is, to have small latency) is the case where the user changes location very frequently, and the exact sequence of visited locations is needed. For instance, verifying if an employee followed the work procedure as expected (e.g., a guard that periodically must check several gates or doors at a factory).

To this end, we created a service for inviolable presence registration of mobile agents. The implementation of the service uses a Distributed Ledger Technology (DLT), which allows us to store and retrieve data while ensuring the integrity of the location information it contains. We incorporated the service into the ContextNet platform [3], an IoMT middleware that already implements two modes of presence detection.

This paper is an extended version of [4]. We have further developed our service to include a control access system for the stored presence data. With this system in place, only the user who owns the data can retrieve the stored location information. We tested the impact of this addition on the service performance and observed that it introduces very little overhead while significantly increasing user privacy.

Moreover, we repeated our previous experiments using a different blockchain called Ethereum [5] and analyzed the differences between this technology and IoTeX [6]. We compared their performances for different settings of mobile entities and clients. Our results show that, overall, Ethereum is more efficient than IoTeX, although presenting a large variance in the measured transaction times.

This text is organized as follows: the "Fundamentals" section presents ContextNet and its presence detection mechanisms and characterizes the concept of blockchain. The "Inviolable presence registration service" section describes our registry service, "Implementation and performance tests" section discusses our experimental results, and "Challenges and opportunities" section discusses challenges and opportunities for the creation of inviolable presence registration systems for mobile entities. The

"Related work" section compares our approach to other studies related to this topic, and the "Conclusion" section has our concluding remarks and possibilities for future work.

Fundamentals

Presence detection in ContextNet

ContextNet [3] is a distributed scalable cloud-mobile-edge middleware composed of the **Core** and the **Mobile-Hub** (or simply M-Hub) [1]. The Core is an extensible set of cloud-based microservices, such as a peer-to-peer, scalable publish/subscribe communication infrastructure. The M-Hub is an extensible microservice framework for Android devices (e.g., phones and low-cost Systems on a Chip (SoCs)) that serves as a connectivity hub for smart IoT devices.

In its current version, the M-Hub provides Internet connectivity to any Bluetooth Low Energy (BLE) device that is within its communication range and supports Complex Event Processing (CEP) [7] as a means of local processing on edge devices. Moreover, it allows the discovery, configuration, and registration of any BLE beacon (e.g., iBeacon/Eddystone) using the web or a remote mobile device. These beacons are devices that periodically broadcast short-range advertisement signals, expecting nearby smart devices (e.g., smartphones) to detect them through a wireless scan and map their identifiers to a meaningful name or coordinate.

To turn the values attributed to beacons into real-world coordinates, we need to assign each of them a symbolic location name or a physical coordinate, typically in a cloud-based external database. Due to their small size and weight, we can embed BLE beacons into wristbands, ID badges, or small tags attached to any equipment (mobile or not), furniture, wall, or ceiling.

In our case, we chose to detect mobile beacons by instrumenting indoor spaces with M-Hubs running on small SoCs (with BLE and Wi-Fi interfaces) attached to ceilings or walls. All M-Hubs have a list of allowed beacon IDs that they will consider while ignoring any other BLE device emitting beacon advertisements for pairing. The BLE technology gives beacons a range of advertisement coverage between 10–50 m, which is dependent on the transmission power of its IEEE 802.15.1 radio. So, to fit the advertisement range of each beacon to our application's needs, we configured each beacon to use a specific transmission power to adjust its reachability. Finally, ContextNet's proximity detection service also performs beacon signal disambiguation based on the radio frequency (RF) signal strength whenever more than one M-Hub detects a beacon. This way, we can build `found()` and `lost()` entries in the mobility log.

In addition to the beacon-based positioning, the ContextNet middleware system also supports mobile entities' positioning using geo-locations and geo-fences. In this case, it uses its GroupDefiner service [8], which allows the developer to specify an arbitrary set of *geo-fences* in the form of a convex polygon of (lat,long)-coordinates. Then, for each geo-reference (e.g., GPS or cell network position) received from the M-Hubs, the GroupDefiner will check to which (one or many) registered polygon the mobile entity belongs and will retrieve the corresponding region-ID.

Complex event processing

To improve the effectiveness and accuracy of presence detection using beacons, we must compare, filter out, correct, and summarize beacon signals. To this end, ContextNet's

M-Hub features a full-fledged ESPER Complex Event engine [9] that can process continuous queries in the form of Event Processing Language (EPL) rules over the stream of arriving beacon signals.

Such continuous queries typically define a sliding observation window of events over which the rules' patterns are checked. For example, a rule can determine that if in the same 15-s window, only one advertisement of strength up to −100 dB is received from a beacon `B1`, then this beacon has moved out of the M-Hub's vicinity. Also using CEP rules, any M-Hub, say `H1`, can map a sequence of subsequent confirmations of a beacon's proximity into initial `found(H1,B1)` and `lost(H1,B1)` events, which are the ones then sent to the service at the Core.

Blockchain

Blockchain is a DLT for recording information into an encrypted chain of blocks so that it can never be modified or forged. Although it has been used as the public ledger for some cryptocurrencies like Bitcoin, it has a wider concept [10].

To employ a blockchain, we must first create a network with all computing nodes interested in using it. When a node executes a transaction, it uniquely signs it, thus ensuring the operation's integrity, and then transmits it to its peers. The transmission block contains valid transactions and references the previous block of the chain using the corresponding hash. If at least one of these conditions is not met, the block is rejected. Otherwise, the nodes add the block to their chain, updating the transactions. This process guarantees the immutability of the data because changing a block requires changing all previous blocks.

There are two types of blockchain: public and private [10]. The main difference between them relates to who is entitled to participate in the network, execute the consensus protocol, and keep the shared data. Public blockchains are entirely open, and anyone can join their networks and access their public transactions. In this case, the network typically has an incentive mechanism to get more participants to use it. Private blockchains, on the other hand, require permission to access the information contained in the chain. Therefore, it is possible to limit the parties allowed to make transactions, be present in the network, and write new blocks in the chain.

As there is no central authority to validate transactions, a blockchain needs a mechanism to reach a decentralized consensus. There are currently two important models for this: Proof-of-Work (POW) and Proof-of-Stake (POS) [10]. Bitcoin [11], Ethereum [5], and almost all cryptocurrencies use the POW model. This model requires computers to "mine" cryptocurrency by solving complex mathematical problems. For every solved problem, these mining computers are rewarded with some cryptocurrency. As an alternative to POW, POS was conceptualized around making mining fairer. It is used by cryptocurrencies such as PIVX [12] and Nxt [13]. In these cryptocurrencies, the system randomly chooses miners, so no energy is wasted by using computers to solve complicated mathematical problems. Consequently, instead of mining, users can invest directly in the coins. Ethereum is in the process of migrating to this system.

In a Delegated Proof-of-Stake (DPOS) system [10], a community of block producers and staked users agree to a specific set of rules to create a technological democracy. Every wallet that contains coins can participate in validating transactions and forming a consensus. Thus, the more coins in a wallet, the more coins it will eventually receive. The wallets can

also vote for representatives, which validate transactions, form a consensus, and are paid for their efforts through the system.

In our previous study [4], we chose IoTeX [6] as the blockchain that stores the location information in our system. Using DPOS, IoTeX is a high-throughput, instant system with reduced transaction costs. Furthermore, it is a privacy-based blockchain with a lightweight system architecture specifically designed for various IoT industries. However, we required additional IoTeX tokens to execute the experiments using the current implementation of our service. Given that the system we used to receive these tokens [14] was not available during the testing period of the new features, we decided to switch to a better-known network with a broader community.

We chose Ethereum [5] for this task, as it is a platform that provides users with the capacity to create their own decentralized applications using Solidity, a language that enables the development of code with varying levels of complexity. Moreover, Ethereum is based on smart contracts, allowing two unknown people to do business without needing a central agent. The terms are programmed in the contract, which is then executed, fulfilling the defined conditions and without human interference, thus reducing the negotiation costs.

Inviolable presence registration service

Presence logs of mobile entities should be capable of being invalidated, as they give information about movement events and traces of the real world. Thus, it is only natural to develop an inviolable presence registration service using blockchain for mobile entities in environments such as rooms, bus stops, shops, among others. To do that, we use M-Hub, the ContextNet component that serves IoMT applications, to interact with sensors and actuators built into smartphones and embedded in smart things. By combining M-Hub with a blockchain network, we ensure that these interactions between smart things can be stored safely and distributedly. With a blockchain-based distributed computing platform and operating system featuring smart contract functionality such as Ethereum, we can also create a control access system for retrieving the stored data.

Either by finding nearby BLE beacons or comparing the geo-location obtained by a GPS with geo-fence limits, the M-Hub is the element of our infrastructure that associates a symbolic location to a mobile entity (either itself or a person/machine passing by with a beacon). Figure 1 shows the possible ways in which we locate/track mobile entities.

The M-Hub uses CEP filters that define when an entity enters or leaves a location or geospatial region. As it is necessary to distinguish each user, their identity has to be protected in some way. We attached the user's identification through the phone device ID. Unlike a user name or identification number, this piece of information is not related directly to the individual. Also, only the device owner would easily have it, making it difficult to associate the mobile device with the user. The phone device ID can also be hashed to make it difficult for others to identify individuals.

The system identifies the current location and waits some time before defining a point as a mobile entity. Initially, we considered using the mean of the positions, but this measure is sensitive to sample values, thus being best suited for situations where data is uniform. Therefore, using the mean could delay or make it impossible to define the mobile entity's location, as it would be necessary to obtain the same point in all samples of that

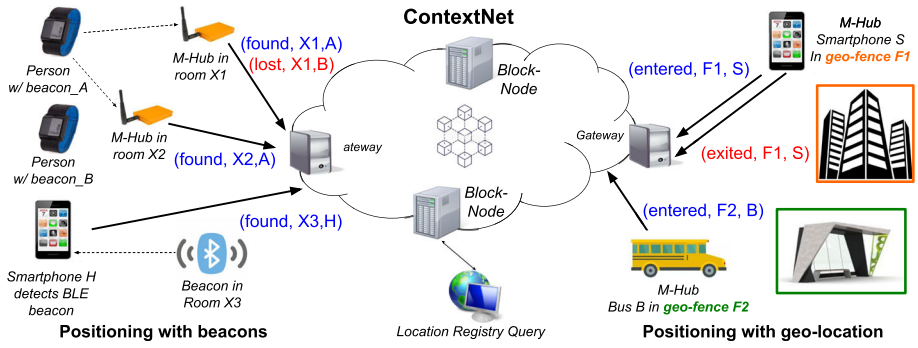

Fig. 1 Immutable presence registration service using blockchain in the Core for both positioning using beacons (left) and geo-locations (right). This figure shows the possible ways in which we locate/track mobile entities

observation period to make sure that it is completely correct. *Mode* [15], in turn, represents the most frequent value of a dataset, and to define it, we observe how often a value appears. By doing so, we analyze the positions received during that period and define the current location by the most recurring. Consequently, we use a continuously calculated mode of the positions, and the system only sends another message if the entity appears in a new filtered position. This way, we reduce the amount of unimportant data sent.

The mobile nodes that contain an M-Hub send this filtered information to a manager node (Fig. 2), which then adds the data to a queue to be sent to a blockchain, thus reducing the latency of the blockchain's process of inserting a new data item. We call this part of our approach, the Block Node (B-Node). Additionally, with this B-Node, we can send location data asynchronously. After an established period, which can vary according to the B-Node's response, this filtered information will be sent to the blockchain, not interfering with the submission of new data. This approach makes the service more agile and allows multiple data to be sent concurrently to the blockchain.

The B-Node was developed to allow the use of various blockchains, as it only defines signatures of methods and properties. Each blockchain implements the practical behavior of the methods. If using another blockchain is necessary, it is possible to do so by changing

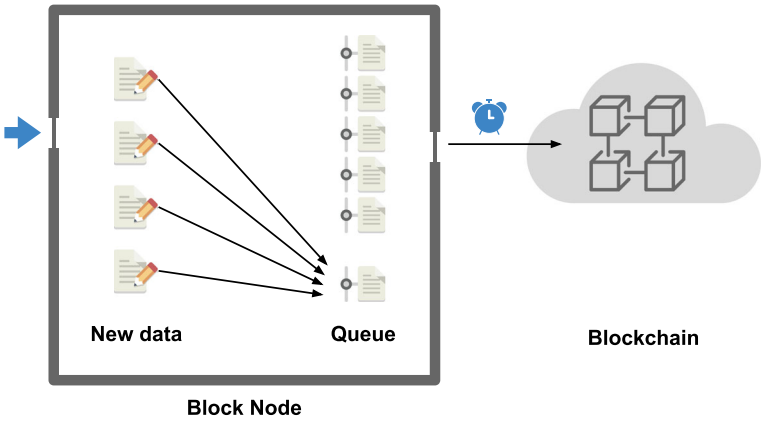

Fig. 2 Block Node sending scheme. This figure shows the steps contained in the Block Node to aggregate and send data to the blockchain

only the component that interacts directly with the blockchain network. The rest of the B-Node would already be ready for use. Furthermore, the B-Node includes a contract that can be used if the chosen blockchain requires a smart contract to interact with the network (e.g., Ethereum).

The B-Node uses the Ethereum blockchain to send multiple messages without suffering lengthy delays. This is possible due to the smart contract mechanism that facilitates the process by allowing messages to be grouped into one before being sent. The smart contract executes the process of identifying messages sent in bulk in a unique identifier. The message sent to the blockchain has the entity identification, the user location, the timestamp, and a message that defines whether the user has left or entered the place. The fact that we are working with individual messages facilitates the search for stored information; however, it is still costly to examine the entire chain looking for the corresponding data.

Written in Solidity, the smart contract code deployed on the Ethereum network contains three methods for interacting with the stored data, as illustrated in Fig. 3. The `setData` method receives as a parameter the String to be sent to the network and the **userID** created from the phone ID hash. It stores this information and generates a unique **transactionID** by combining the user ID and the current time of the transaction. We also create a link between **userID** and **transactionID**, thus shaping an access control system where only the owner of the data can retrieve their own location information. We prefer to store the data using hash maps since this structure reduces the cost of searching for the stored data in `getData`. We also created the `getTransactions` method to return all the stored transaction IDs for a specific user. The user sends their ID, and the smart contract generates the corresponding hash. We use this hash to retrieve the data, and with that, we can control who accesses the stored information.

Combining the B-Node service with the ContextNet middleware provides a way to identify an entity's presence at a place or region and measure how long it stayed at a place. This information is sent via M-Hub to a blockchain network, thus ensuring the immutability of the information. As a result, it is possible to build location-based services that register the position and the period that a mobile entity, such as a robot, drone, or human, stayed in that location.

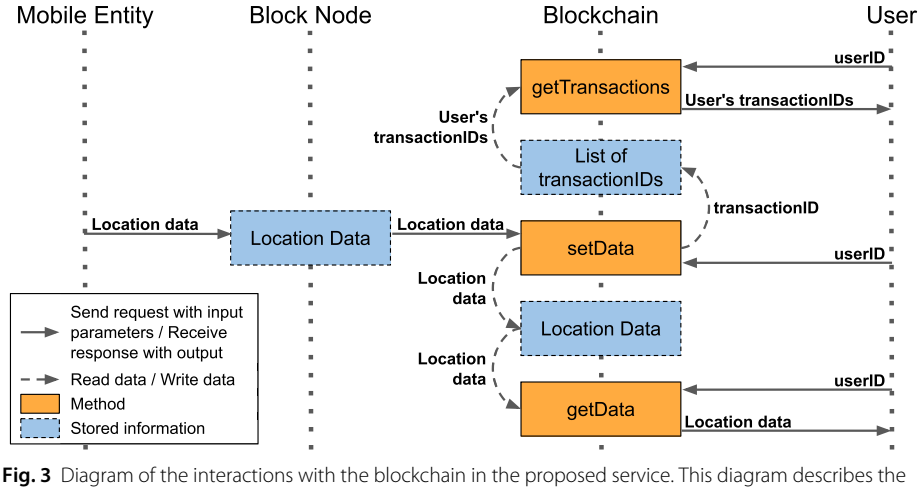

Fig. 3 Diagram of the interactions with the blockchain in the proposed service. This diagram describes the interactions between the mobile entity, the Block Node, the blockchain, and the user

Implementation and performance tests

The experiments described in this section aim to explore the concept presented in this paper by observing the efficiency of identifying an entity's position and sending, adding, and retrieving this stored inviolable information.

Experimental setup

In our experiments, we measured the latency of the service transactions on a local 280-megabit network. Except where stated otherwise, the reported results represent the average transaction time of 5 executions for IoTex and 30 executions for Ethereum, along with their respective standard deviations and margins of error calculated with a confidence level of 95% using Microsoft Excel 2013. This disparity in the number of executions is due to the difficulty in obtaining more tokens for the IoTeX transactions, as explained in the " Blockchain" section. For the communication between the mobile entities and the Block Node, we used version 2.7 of the Contextnet middleware.

We employed an indoor location dataset collected by PUC-Rio students and professors that contains the time series generated by beacons deployed in real-world office environments. To simulate the movement of mobile entities, we used a mobile node (MN) and a stationary node within the ContextNet Scalable Data Distribution Layer (SDDL) core network architecture [3]. This stationary node of the SDDL core acted as a server processing node capable of processing application messages from the MN (according to an application-specific logic) and sending messages back to the mobile node.

This information is sent to the Ethereum Testnet. This Testnet is an alternative Blockchain that acts as a global testing environment in which developers can obtain and spend ether with no real value on a very similar network to the regular Ethereum network (also called Mainnet). Although similar, a Testnet has fewer validation nodes compared to the regular network. This kind of alternative to the Mainnet allows developers to experiment with new code and solutions. In doing so, they are not "disturbing" the regular network, nor are they forced to use tokens that have real value. We decided to use it instead of the Mainnet because it allows us to run tests on Ethereum without worrying about the amount we would be spending. To use a Testnet, we need to create a wallet and obtain free tokens to send a smart contract that works as an interface between the queue and the blockchain. This contract, written in Solidity, has two functions: adding the new data and getting the stored information. With this structure, we were able to send location data asynchronously to the blockchain.

In the IoTeX experiments, we sent the information to the official IoTex Testnet [16], and we used the official SDK, IoTeX Antenna [17], written in Java, to facilitate the smart contract deployment and the interaction with its functions. In the case of the Ethereum experiments [18], we sent the information to Ropsten, which is an Ethereum Testnet [19], and we used the official software development kit, Web3j [20], also written in Java.

IoTeX vs. Ethereum

By changing the employed blockchain from IoTeX to Ethereum, we were able to execute our previous test scenarios [4] using this technology and compare the two approaches.

In the first experiment, we varied the waiting period in the queue that retains the location data before sending them to the blockchain. Starting at 5 min and increasing the waiting time by increments of 1 min, our goal was to study the latency of a transaction

that inserts a new registry. To speed up the process, we sent a new message every 15 s and increased the transaction time until we reached a 12-min wait time. We chose not to use the access control system to see how the B-Node would perform with the increase of data and provide a fairer comparison between the two tested blockchains. Figure 4 shows these results.

From the graph, we can see that while IoTeX's average transaction time grows logarithmically with the wait time, Ethereum's stays approximately constant. However, due to the large variance in the times obtained for Ethereum, the difference only starts to be statistically significant from the point of an 8-min wait time onward. The fact that only the Ethereum tests present a large number of outliers indicates that this issue is related to the performance of the Ethereum Testnet rather than to the proposed service.

Based on these results, we can consider that the insertion operation is efficient in both blockchains, with Ethereum being the most suitable approach for cases with large wait times and IoTeX being the most suitable option for cases where the wait time is below 8 min and more predictable execution time is required.

In the second experiment, we fixed a 5-min waiting period. We increased the number of mobile nodes that are concurrently trying to insert a new registry in the system to analyze how this would affect this operation's latency. Figure 5 illustrates these results.

Once again, we can consider both blockchains to be efficient, as IoTeX's transaction time grows slightly below linear-scale, and Ethereum's continues to be approximately constant. Nonetheless, in this scenario, only the results for the case where one mobile node is executing the operation are statistically similar, with Ethereum outperforming IoTeX by 1.61 to 3.13 times when more mobile nodes are employed.

We note that the process of transaction verification used by Ethereum for the insertion operation helps justify its better performance. With this mechanism, there is no need to

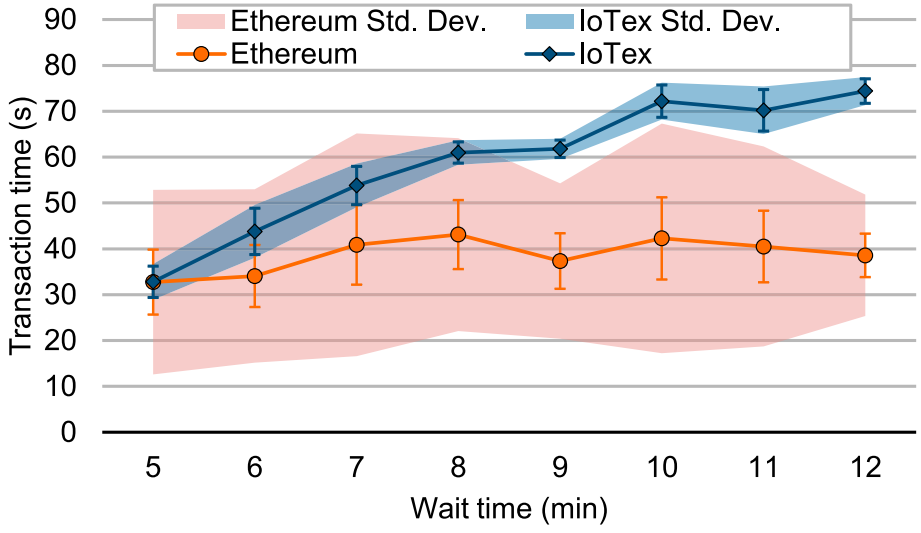

Fig. 4 Comparison between IoTeX and Ethereum average transaction times for different wait times (lower is better, error bars were calculated with a 95% confidence interval). This graph compares the average transaction time of the IoTex and Ethereum implementations for a scenario where the waiting time on Block Node is being increased

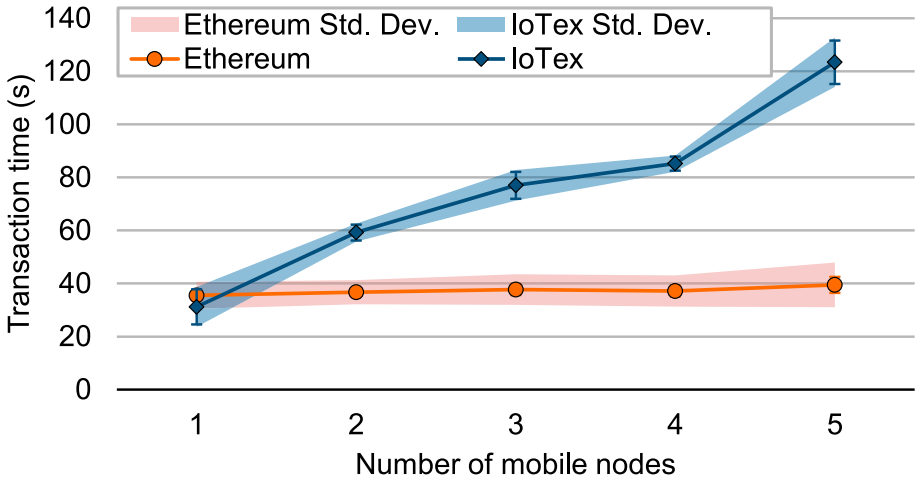

Fig. 5 Comparison between IoTeX and Ethereum average transaction times for a different number of mobile nodes (lower is better, error bars were calculated with a 95% confidence interval). This graph compares the average transaction time of the IoTex and Ethereum implementations for a scenario where the number of mobile nodes is being increased

wait for the transaction to be completed, as a local node validates the transaction. Moreover, the sender is given a hash that serves as a guarantee that the transaction will be confirmed by the network later.

Access control system performance

In our second set of experiments, we evaluated the performance of the new feature we implemented in our service, the access control system.

In the first experiment, we measured the time of a data insertion transaction with and without the access control mechanism. We set a 10-min waiting period in both cases. We observed an average transaction time of 42.27 s (with a $\pm$ 8.96 s margin of error) without the access control system and of 40.37 s ($\pm$6.71 s margin of error) with it. Therefore, there was no statistically significant impact on the average transaction time due to the mechanism's introduction, indicating the efficiency of the function defined in the smart contract.

Figure 6 is a histogram showing how the results are distributed in 20-s intervals. By analyzing the histogram, we see that the interval with most results obtained without the system (30%) was between 0 s (inclusive) and 20 s (non-inclusive), while the interval with most results with the system (33%) is 20 s (inclusive) and 40 s (non-inclusive). Given the high variability of Ethereum insertion transaction times, this indicates that the control access system introduced very little overhead to the transaction times.

In the second experiment, we tested the access control system's impact on the transaction times of a data retrieving task. With five executions of the task with and without the mechanism, we observed the same distribution of values (always either 4 or 5 s). Again, this indicates that the mechanism did not introduce significant overhead to our service.

The short transaction time compared to the insertion task is due to how we stored the data. We employed a hash map where the **transactionID** is the key used to retrieve the information. Hash maps tend to present good performance when storing large sets of

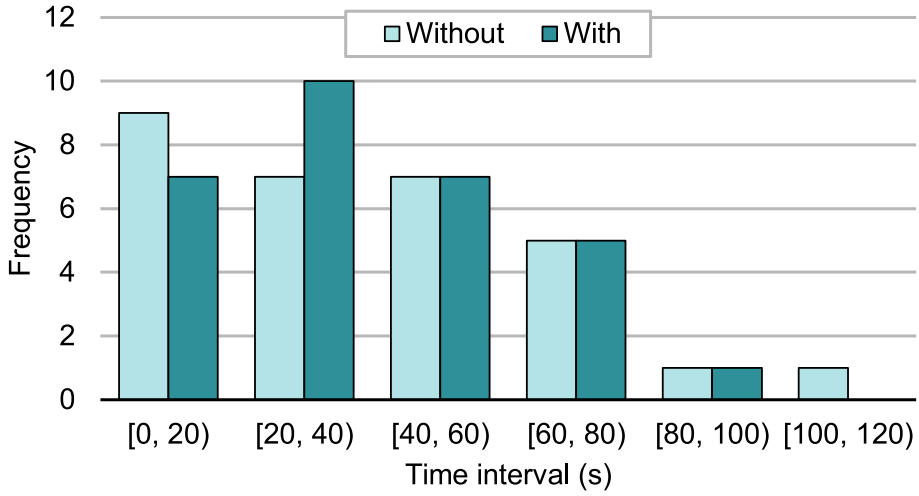

Fig. 6 Histogram of Ethereum transaction times with and without the access control system. This graph compares the impact on the transaction time of inserting new data in a system with and without the access control system

data, especially when the keys are unknown until the run time or when dealing with a single type of key and a single type of value.

In the third experiment, we analyzed our service's performance, including the access control mechanism, where there are multiple simultaneous requests to retrieve data. We tested for 10, 20, and 30 simultaneous accesses, and then from 100 to 500 accesses in increments of 100 users. Figure 7 illustrates these results.

We see that there is no statistically significant difference between the results, with average transaction times ranging between 4.7 and 6.1 s, indicating that the service scales well for multiple simultaneous consults. This result shows how a decentralized approach

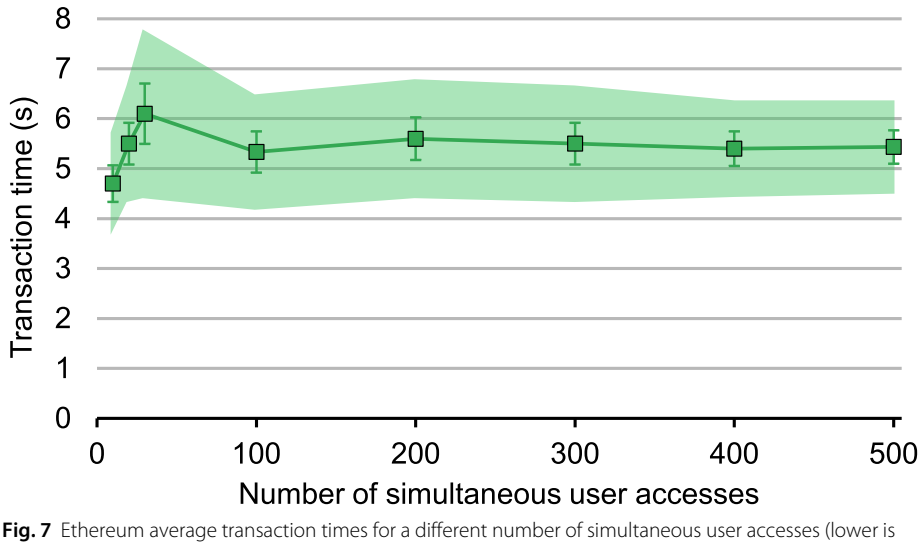

Fig. 7 Ethereum average transaction times for a different number of simultaneous user accesses (lower is better, shaded area represents the standard deviation, error bars were calculated with a 95% confidence interval). This graph compares the impact on Ethereum average transactions caused by increasing the number of users simultaneously consulting data

is useful for aspects beyond security. Each node in the Ethereum network has a copy of the state, and each node independently verifies the state's authenticity as blocks arrive and transactions are processed locally. The `getData` method is a view function that, by definition, is read-only. These types of functions ensure that the state is not changed. As there is no attempt to change the state, the network is not even consulted during these calls. Therefore, any node can explore and inspect the state without any additional assistance from the network, significantly reducing the transaction time, given that no mining is required.

Challenges and opportunities

Security

The basic concept of blockchain is that every transaction is recorded in blocks. This transparency has enabled several cybercriminals to be caught, as it is possible to track the history of transactions that are forever written in the blockchain. Still, the transparency of this data is not always desired due to privacy concerns. For example, in Europe, the General Data Protection Regulation (GDPR) states that everyone has the right to revoke their consent at any time and permanently access or delete all types of information they have previously agreed to share. This raises the question: how does this fit in with the permanent blockchain registry? Once data has been added to the chain, it cannot be removed or changed.

In this context, blockchain-based services must create mechanisms to protect their users' data. In our case, the system controls access by using the personal device IDs to identify and store the mobile entities. To not suffer from improper exposure, the user's identity must be stored carefully.

Positioning accuracy

GPSs and other Global Navigation Satellite Systems face difficulties typical of any radio broadcast, given that being between large buildings and metallic structures can disturb the reception of the signal or a smartphone's ability to get the location information. Also, a low-quality device with a slow processor or weak receiver can significantly disrupt the navigation. Beacons can also present problems, as there may be momentary failures while connecting to mobile entities. For a system that needs the location of entities to identify and send the data to the blockchain, these problems can affect the stored information's reliability.

Scalability

The scalability of a system is its ability to handle additional workloads. However, blockchain networks are generally slow and inefficient [21]. The classic blockchain is not scalable, as increasing the number of nodes in the network does not increase the throughput of transactions. As a way to speed up this process, the Lightning Network [22] was invented. This technique implements a smart contract script on the network that opens private payment channels between one party and all other parties with which they traded. In addition to the private payment channels, each party has an open channel for the original blockchain. The parties can thus trade with each other using private channels, and only the final outcome of the transaction is passed to the blockchain. Nevertheless, this solution cannot be applied to the

creation of inviolable presence registries, given that every transaction is essential in this scenario.

Related work

Victor and Zickau [23] propose that geo-fences be defined in smart contracts as part of Location-based Services (LBS) and that current geographical positions provided by mobile users be evaluated on whether they are contained in the geo-fence or not. Their approach utilizes well-established location encoding systems (e.g., geo-hashes and S2 cells) that transform polygons into a grid of cells, which in turn are stored in the smart contract as a representation of the geo-fence. The authors evaluated these two location-encoding systems in terms of the storage and processing costs in an Ethereum-based smart contract implementation, identifying that the codification as S2 cells is much more efficient. Using geo-fences to identify one's presence is a common aspect between our study and theirs. However, we store the already validated positioning data instead of developing a smart contract to evaluate whether the position is in the geo-fence or not.

Brambilla, Amoretti, and Zanichelli [24] propose an approach for producing a proof of location system, which are digital certificates that attest to someone's presence in a particular geographic area. The authors identified that centralized verification approaches proposed in the past are not satisfactory, as they can be a high risk to the user's privacy. Their paper illustrated a completely decentralized, blockchain-based scheme that guarantees location integrity and preserves user privacy. We also share the idea of a blockchain-based system to prove the location of an entity. Nonetheless, we do not include and share a chain with the data between the nodes, given that we use an independent public network with the M-Hub connections.

Furthermore, Brambilla, Amoretti, and Zanichelli consider an LBS peer-to-peer network with mobile nodes that are connected to the Internet and can interact with neighboring nodes through short-range Bluetooth communication. In their paper, they define two roles for the nodes in the network: prover and witness. A prover is a node that wants to collect proofs of location from its neighbors, while a witness is a node that has provided proof of location to the prover. Every peer is described by a unique identifier, its public key, and can digitally sign information with its identifier's private key.

Conclusion

This paper discusses the need for a middleware-level service for inviolable presence registration of mobile entities. We present an approach to implementing it using a blockchain as a Core service of the ContextNet middleware, taking advantage of its CEP capabilities at the edge (i.e., the Mobile-Hubs) and the cloud (i.e., the Core). This type of service's performance is the cornerstone of the viability of some real-world mobility monitoring and tracking applications, such as the one we are developing in our departmental rooms and offices. Therefore, our experiments evaluate the efficiency of tasks such as storing and retrieving inviolable information.

We have further developed our service compared to the initial minimum viable product to include a control access system for the stored presence data. With this mechanism in place, only the user who owns the data can retrieve the stored location information. We observed that the mechanism introduced very little overhead to our service's insertion

and retrieval tasks. This is partly due to the data structure chosen, as hash maps allow us to access the data directly if we have the permission and the **transactionID**.

We have also changed the blockchain technology employed from IoTeX to Ethereum, as the more accessible tokens for the Ethereum Testnet allows us to execute a larger number of experiments. Our results indicate that, due to blockchain technologies' current characteristics and limitations, they can only be used for medium-sized presence registration applications, that is, only for a few hundred users with updates every 39 s or more. Improving the service's efficiency and increasing data registration capacity are essential steps that we will explore in the future.

We point out that, although storing data in a blockchain is a cumbersome process with potential high delay, there are ways to avoid these transaction time disruptions in the flow of new data, such as putting these data in a queue before adding them to a blockchain. The queue also allows us to manage how long we wait before sending a message, given that it is possible to change this period. However, this process can have a negative impact on the overall message sending (and data recording) time delay, as the wait time in the queue impacts many messages, leading to each message's wait time to be higher. Considering the latency in a larger scale installation that includes the cloud and a message queue to deliver the messages should be among the main next initiatives to be addressed.

Abbreviations

BLE: Blue-tooth Low Energy; CEP: Complex Event Processing; DLT: Distributed Ledger Technology; DPOS: Delegated Proof-of-Stake; EPL: Event Processing Language; GDPR: General Data Protection Regulation; IoMT: Internet of Mobile Things; IoT: Internet of Things; LBS: Location-based Services; MN: Mobile node; POS: Proof-of-Stake; POW: Proof-of-Work; RF: Radio frequency; SDDL: Scalable Data Distribution Layer; SoC: System on a Chip

Acknowledgements

The authors thank the reviewers of the 9th Latin-American Symposium on Dependable Computing (LADC 2019) for their feedback and suggestions.

Authors' contributions

All authors read and approved the final manuscript. They also were responsible for the analysis and interpretation of data.

Funding

This work is supported in part by FAPESP (grant #2019/19312-9).

Availability of data and materials

An anonymous version of the data can be made available upon request.

Competing interests

The authors declare that they have no competing interests.

Received: 27 May 2020 Accepted: 3 January 2021
Published online: 21 January 2021

References

1. Talavera LE, Endler M, Vasconcelos I, Vasconcelos R, Cunha M, da Silva e Silva FJ (2015) The Mobile Hub concept: enabling applications for the Internet of Mobile Things. In: Pervasive Computing and Communication Workshops (PerCom Workshops). pp 123–128. https://doi.org/10.1109/PERCOMW.2015.7134005
2. Nahrstedt K, Li H, Nguyen P, Chang S, Vu LH (2016) Internet of Mobile Things: mobility-driven challenges, designs and implementations. In: First IEEE International Conference on Internet-of-Things Design and Implementation (IoTDI). pp 25–36. https://doi.org/10.1109/IoTDI.2015.41
3. Endler M, e Silva FS (2018) Past, present and future of the ContextNet IoMT Middleware. Open J Internet Things (OJIOT) 4(1):7–23
4. Leal M, Pisani F, Endler M (2019) Inviolable presence registration of mobile entities in the ContextNet Middleware. In: 9th Latin-American Symposium on Dependable Computing (LADC). pp 1–4. https://doi.org/10.1109/LADC48089.2019.8995684
5. Buterin V, et al. (2014) A next-generation smart contract and decentralized application platform. https://github.com/ethereum/wiki/wiki/White-Paper. Accessed 29 Apr 2020
6. IoTeX (2019) The Internet of Trusted Things. https://medium.com/iotex/the-internet-of-trusted-things-a29a0e4f169. Accessed 07 May 2020

7. Luckham DC (2001) The power of events: an introduction to complex event processing in distributed enterprise systems. Addison-Wesley Longman Publishing Co., Inc., Boston. https://doi.org/10.1007/978-3-540-88808-6_2
8. Vasconcelos RO, Silva LDN, Endler M (2014) Towards efficient group management and communication for large-scale mobile applications. In: Pervasive Computing and Communications Workshops (PERCOM Workshops), IEEE International Conference On. pp 551–556. https://doi.org/10.1109/PerComW.2014.6815266
9. Talavera LE, Endler M, Colcher S (2016) An Energy-aware IoT Gateway, with continuous processing of sensor data. In: Proc. of the Brazilian Symposiym for Computer Netwoks and Distributed Systems (SBRC). pp 1083–1096. http://www.sbrc2016.ufba.br/downloads/SessoesTecnicas/152343.pdf
10. Zheng Z, Xie S, Dai H, Chen X, Wang H (2017) An overview of blockchain technology: architecture, consensus, and future trends. In: IEEE International Congress on Big Data (BigData Congress). pp 557–564. https://doi.org/10.1109/BigDataCongress.2017.85
11. Nakamoto S (2008) Bitcoin: a peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf. Accessed 29 Apr 2020
12. Strontium (2018) PIVX White Paper. https://pivx.org/wp-content/uploads/2019/05/PIVX-White-Paper-Sept-2018.pdf. Accessed 07 May 2020
13. NxtCrypto (2014) What is Nxt? http://www.nxtcrypto.org/nxt-technology/what-nxt. Accessed 07 May 2020
14. IoTeX Team (2018) IoTeX: a decentralized network for Internet of Things powered by a privacy-centric blockchain. https://whitepaper.io/document/131/iotex-whitepaper. Accessed 11 May 2020
15. Springer Verlag GmbH EuropeanMathematicalSociety Encyclopedia of mathematics. https://www.encyclopediaofmath.org/. Accessed 29 Apr 2020
16. IoTeX Testnet IoTeX Scan. https://testnet.iotexscan.io/. Accessed 08 May 2020
17. IoTeX iotex-antenna. https://iotexproject.github.io/iotex-antenna/. Accessed 08 May 2020
18. Lee W-M (2019) Connecting to the Ethereum blockchain. Apress, Berkeley. https://doi.org/10.1007/978-1-4842-5086-0_3
19. ropsten ropsten. https://ropsten.etherscan.io/. Accessed 08 May 2020
20. Web3 Labs Where Java meets the blockchain. http://web3j.io. Accessed 08 May 2020
21. Dogan T (2019) Who scales it best? Blockchains' TPS analysis. https://hackernoon.com/who-scales-it-best-blockchains-tps-analysis-pv39g25mg. Accessed 29 Apr 2020
22. Poon J, Dryja T (2016) The Bitcoin lightning network: scalable off-chain instant payments. https://lightning.network/lightning-network-paper.pdf. Accessed 29 Apr 2020
23. Victor F, Zickau S (2018) Geofences on the blockchain: enabling decentralized location-based services. IEEE Int Conf Data Min Workshops (ICDMW):97–104. https://doi.org/10.1109/ICDMW.2018.00021
24. Brambilla G, Amoretti M, Zanichelli F (2016) Using block chain for peer-to-peer proof-of-location. arXiv preprint arXiv:1607.00174. https://arxiv.org/pdf/1607.00174.pdf

Publisher's Note