A Secure Semantics-enhanced Decentralized Open IoT Service Platform

Vikram Pandey
Department of Computer Science
North Dakota State University
Fargo, USA
vikram.pandey@ndsu.edu

Juan Li
Department of Computer Science
North Dakota State University
Fargo, USA
j.li@ndsu.edu

Yan Bai
School of Engineering and Technology
University of Washington Tacoma
Tacoma, USA
yanb@uw.edu

***Abstract*— This paper proposes an innovative IoT service platform that leverages blockchain technology to enable secure and scalable IoT services. The platform is designed to be open to multiple IoT services and applications, and can be easily extended to provide secure registration, discovery, access, and payment services to both IoT providers and consumers. The proposed platform adopts an ontology-based approach for describing IoT services, which enables flexible and natural search for relevant services by both humans and machines. The use of semantic service descriptions also allows for effective access control to device data and secure transactions through the use of smart contracts. To increase scalability, we have utilized a peer-to-peer (P2P) based method to store and index service metadata, which enables efficient and scalable service discovery. Our experimental results have demonstrated the effectiveness of the proposed system in providing secure and scalable IoT services, while also enabling flexible and natural search for relevant services. Overall, the proposed IoT service platform has the potential to make IoT more accessible for daily use, while also providing enhanced security and scalability for IoT services.**

Keywords—Internet of Things, blockchain, service, ontology, security, smart contract, access control

I. INTRODUCTION

The growth of the Internet of Things (IoT) has led to the deployment of millions of IoT devices across various domains, including homes, hospitals, laboratories, factories, and cars. However, with the increasing number of IoT devices, discovering and consuming IoT services in a secure, flexible, and efficient manner has become a significant challenge[1]. Numerous IoT platforms are available for consumers to access IoT services, such as smart home platforms like Google Home and Amazon Alexa, which provide a user interface for connected devices. Healthcare IoT platforms, like GE Health Cloud [2] and IBM Watson Health [3], offer medical professionals access to patient data and analytics, while industrial IoT platforms like Siemens MindSphere [4] and General Electric's Predix [5] provide real-time data monitoring and analysis in manufacturing and industrial settings.

However, current IoT service platforms are typically designed for single domains, where project-specific policies and requirements are predefined, and all platform components are tightly coupled. This limitation makes it difficult to extend and integrate IoT services, hindering the platform's ability to support new business and applications.

This paper presents an innovative solution to overcome the challenges faced by current IoT service platforms. The proposed platform provides a flexible and open environment, allowing easy integration and extension of IoT services for multiple domains. It offers secure IoT registration, discovery, access, and payment services to both providers and consumers. The platform employs ontology-based descriptions to facilitate both human and machine understanding of IoT services, while smart contracts ensure encryption, access control, and secure transactions. Additionally, the platform leverages a decentralized peer-to-peer network for service registration and discovery, ensuring scalable and robust services. By using blockchain technology, the platform offers a structured, logical, and secure ecosystem for IoT devices, making IoT services accessible and consumable like any other daily service.

The proposed platform has undergone comprehensive evaluations through use case studies, simulations, and emulation experiments, revealing that it functions as expected. In essence, this paper offers a novel solution to the current limitations of IoT service platforms by providing an open, extensible, and secure ecosystem that easily integrates IoT devices into new business and applications.

II. RELATED WORK

In recent years, there has been a growing interest in developing IoT service platforms. Here, we review some of the most relevant works in this area.

One of the most popular approaches for building IoT service platforms is based on centralized cloud architectures [6][7][8]. For example, in [9] Taherkordi et al. proposed a generic cloud-based IoT service access model for smart cities. They structure the description of IoT services in a hierarchical model and populate them in a tree structure containing pointers to services and their corresponding data. The cloud-based tree ensures scalable and fast service provisioning. To improve the security of cloud based IoT service platforms, Li et al. proposed a platform that include a trust assessment framework for security and reputation of IoT services [10]. The trust assessment method based on security employs security metrics that are specific to

This work was supported by the National Science Foundation (NSF) with award numbers: 1722913, 2218046 and 1921576.

the cloud in order to evaluate the security of a cloud service. Additionally, the trust assessment method based on reputation utilizes feedback ratings on the quality of cloud service to evaluate the reputation of a cloud service. These platforms offer a range of features for device management, data processing, and visualization. However, these platforms have limitations in terms of scalability, flexibility, and privacy.

To overcome the limitations of centralized architectures, a number of decentralized approaches have been proposed. For instance, some works propose to use distributed ledger technologies, such as blockchain, to create decentralized IoT service platforms [11][12][13]. For example, in [14] the authors proposed a decentralized name resolving system for IoT services. They construct the IoT name resolving database on the blockchain leger. The system incorporates multiple name resolving schemes to facilitate the adoption of existing IoT naming schemes, and delegates the responsibility of locating individual IoT devices to the respective IoT service providers. While in [15], the authors propose DeTEC, a decentralized and trusted edge computing platform that provides a unified interface to users, resolves requests to the most appropriate server through a domain name server, and utilizes blockchain technology for accountability and rewards. These platforms enable users to register and discover IoT devices and services in a more secure and transparent way. However, they often suffer from low scalability due to the high computational overhead of the blockchain consensus mechanisms. In addition, they cannot support search or query of services based on complex service descriptions.

Another line of research focuses on using semantic web technologies to enhance the interoperability and understanding of IoT data [16][17][18]. These works propose to use semantic ontologies to describe IoT devices and services in a machine-readable way. This approach can improve the searchability and reuse of IoT data. However, due to the complexity of semantic description, these service platform need to be run on a centralized server, which may cause the security and scalability issue.

In spite of the numerous IoT service platforms available, there remains a demand for a more open and decentralized platform that offers advanced security and semantics to enable efficient discovery and consumption of IoT services.

III. System Design

A. System Overview

The proposed system architecture consists of multiple components including IoT devices, fog nodes, and end users.

- IoT devices: Devices that are equipped with sensors, processors, and communication modules that allow them to collect and transmit data to other devices or systems. They provide a wide range of services such as monitoring, control, automation, and analytics. These services can be used by end users or other IoT devices for integrated services. IoT devices are typically connected to the system through wireless connections such as Wi-Fi, Bluetooth, ZigBee, or cellular networks, and they have limited battery power, computing and storage capacity.
- Fog nodes: Fog nodes are computing and storage nodes located at the edge of the network, closer to IoT devices. They provide computational and storage resources for IoT devices, and can act as gateways to connect IoT devices to the blockchain or other network resources. In a blockchain network, fog nodes act as full nodes, which means they maintain a complete copy of the blockchain ledger, validate transactions and blocks, and participate in the consensus mechanism. In addition to their role as blockchain full nodes, fog nodes also index store service descriptions from other IoT devices. These service descriptions are represented using ontologies, which provide a machine-understandable description of the service, including its inputs, outputs, and capabilities. By storing service descriptions, fog nodes can act as intermediaries between IoT devices and end-users, allowing them to discover, select and access services provided by other devices in the network.
- End users: End users are individuals or organizations who consume or utilize the IoT data and services stored on the blockchain. They interact with the blockchain network through light nodes, which are typically mobile or web applications that can connect to the network and perform read-only operations such as accessing data and verifying transactions. End users can also request new services or register their own devices on the network by interacting with the smart contract deployed on the blockchain. The smart contract facilitates interactions between end users and IoT devices, ensuring secure and efficient access to services while eliminating the need for a trusted third party.

This architecture enables the creation of a fully decentralized platform that eliminates the need for a centralized cloud server, which can be a security risk and scalability issue. It ensures secure, efficient, and flexible access to IoT services.

B. Ontology-based Service Description

We designed an ontology for IoT devices and services to improve interoperability in distributed environments. This ontology is a formal representation of the concepts, entities, and relationships that define the structure and behavior of IoT service[19]. It provides a standardized vocabulary for describing and categorizing IoT services and their associated data, functions, and features. We employ IoT-Lite [20], a lightweight instantiation of the semantic sensor network (SSN) ontology to describe the key IoT concepts and relationships. It includes classes to represent IoT devices, sensors, actuators, and other components, as well as properties to describe their attributes and behaviors. Additionally, the ontology defines relationships between services, such as composition or aggregation, and provide mechanisms for service discovery, composition, and management[21][22][23]. Fig. 1 shows part of the proposed ontology.

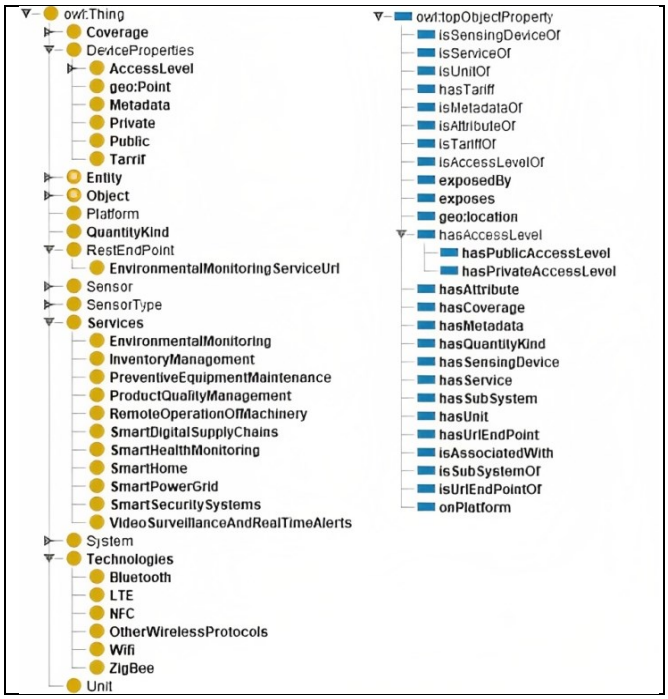

Fig. 1 The high level ontology

An IoT service example is shown below.

```
@prefix : < http:// https://cs.ndsu.edu /iot-onto# > .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>

:Sensor1 rdf:type owl:NamedIndividual ;
  rdf:type :TemperatureSensor ;
  :hasAccessLevel :Public ;
  :hasQuantityKind :Temperature ;
  :hasUnit “Celcius"^^xsd:string ;
  :exposedBy :EnvironmentalMonitoringService ;
  :locatedIn :MonitoringRoom1 ;
  :hasSerialNumber "12345"^^xsd:string ;
  :hasMeasurementRange "−40 to 125 °C"^^xsd:string ;
  :hasAccuracy "±0.5 °C"^^xsd:string ;
  :hasResolution "0.0625 °C"^^xsd:string .

:MonitoringRoom1 rdf:type owl:NamedIndividual ;
  rdf:type :Room ;
  :hasLatitude 46.92^^xsd:decimal ;
  :hasLongitude  -96.81^^xsd:decimal.
```

In this example, the service is a temperature sensor that measures temperature in Celcius. The sensor is kept in a Monitoring Room called “MontoringRoom1” and the location is also described using the latitude and longitude. This ontology provides a complete description of a temperature sensor IoT service, including its unique identifier, location, measurement unit, and sampling interval. It is represented in Web Ontology Language (OWL). Now assume an end user wants to find a publicly accessible IoT service that provides temperature observations at a specific location. A query written in SPARQL is as follows:

```
SELECT ?sensor
WHERE {
  ?sensor a :TemperatureSensor .
  ?sensor :hasAccessLevel ?accessLevel .
  ?sensor :hasQuantityKind ?measures .
  ?sensor :exposedBy ?service .
  ?service a :EnvironmentalMonitoring .
  ?sensor geo:location ?point .
  ?point :hasLatitude ?lat .
  ?point :hasLongitude ?lon .
  FILTER (
    ?accessLevel = :Public &&
    ?measures = :Temperature &&
    ?lat >= 46.00 && ?lat < 47.00 &&
    ?lon <= -96.00 && ?lon > -97.00
  )
}
```

Overall, an IoT service ontology aims to enhance interoperability and enable more efficient and effective development, deployment, and discovery of IoT applications and services.

C. Blockchain Network Formation

By forming a blockchain network with fog nodes as full nodes, the platform enables service registration/verification, service access, and user authentication to be performed through smart contracts. Fog nodes can also participate in the consensus process to validate transactions and add new blocks to the blockchain. This architecture enables the creation of a fully decentralized platform that eliminates the need for a centralized cloud server, which can be a security risk and scalability issue. Fog nodes can help to distribute the computation and storage tasks that would otherwise burden the cloud, resulting in a more efficient and scalable system. The main functions of a full node in a blockchain network include:

- Validating transactions: Full nodes verify and validate every transaction added to the blockchain network, ensuring that they comply with the network's rules and protocols.
- Propagating transactions: Once a transaction is validated, full nodes broadcast it to other nodes on the network, helping to ensure that the transaction is included in the next block.
- Maintaining a copy of the blockchain: Full nodes store a complete copy of the blockchain ledger, which enables them to validate new transactions and blocks, as well as maintain a history of all previous transactions.
- Mining blocks: Some full nodes participate in the process of adding new blocks to the blockchain, known as mining. As fog nodes may have limited resources, they are designed to mine blocks in a more energy-efficient manner by using proof-of-stake (PoS) consensus algorithms[24]. In PoS, nodes are selected to validate transactions and create new blocks based on the amount of cryptocurrency they hold (i.e., their stake) instead of solving complex mathematical problems as in proof-of-work (PoW). This eliminates the need for energy-intensive mining hardware and reduces the energy consumption of the network.
- Enforcing network rules: Full nodes help enforce the network's consensus rules by rejecting any blocks or transactions that do not comply with the established rules.

D. Smart Contract-enabled Service Operations

Smart contracts have been employed to enable secure and efficient IoT service operations[25]. With smart contracts,

service providers and consumers can interact with each other directly without the need for intermediaries, such as centralized platforms or third-party payment processors. This results in reduced transaction costs, increased transparency, and improved trust between service providers and consumers. Smart contracts are created and deployed on the blockchain network, which is accessible by the fog nodes and other blockchain full nodes. The end-users and IoT owners access the smart contract using clients, which can be either web-based or mobile-based. These clients interact with the smart contract using its function.

The smart contract functions can be executed by the users to perform different tasks such as registering their devices/service on the network, requesting access to the IoT device data, paying for the services. The fog nodes in the network act as full nodes, which store a complete copy of the blockchain and execute the smart contract functions on behalf of the users. This reduces the workload on the end-user devices and provides a secure environment for executing the smart contract functions. After executing the smart contract function, the fog node generates an authorization token for the user, which can be verified off-chain by sending a hash of the token and the private and public keys to the access point.

In our system, smart contracts are used to enable user authentication, service registration and verification, service access, and secure payment.

- User registration: The user registration function allows users to securely register and authenticate their identities, enabling them to access the network's services. It requires certain user details, such as a public key and other relevant personal information. Once the registration is complete, the smart contract can generate a unique user ID and record it on the blockchain, creating an immutable and transparent record of the user's identity. The user ID can be used to authenticate the user for subsequent network access requests.
- Service registration: The service registration function enables service providers to register their IoT services. It requires service providers to provide details about their services, such as service type, location, access policy, and service fees. The smart contract then generates a unique service ID and records it on the blockchain, creating a tamper-proof and immutable record of the service's availability.
- Service access: Service access requires that a user provide a valid authentication token before accessing a service. The smart contract can then check that the token is valid, and that the user has the necessary permissions to access the service. Additionally, service providers can leverage smart contracts to validate their financial capacity to pay for the service. Service access can be based on various access control policies, such as role-based access control, attribute-based access control, and blockchain-based access control. Once the user is authenticated and authorized to access the service, the smart contract can execute the service and record the transaction on the blockchain.
- Service payment: When a user requests a service, the smart contract can automatically verify the user's account balance, ensuring that they have sufficient funds to pay for the service. If the user has enough funds, the smart contract can execute the payment and record the transaction on the blockchain, generating a unique payment ID for future reference. The service payment function can also incorporate features such as escrow and dispute resolution mechanisms to mitigate any potential conflicts or disputes between users and service providers. For example, the smart contract can hold the payment in escrow until the service has been fully delivered, ensuring that both parties are satisfied with the transaction before releasing the funds.

Fig. 2 shows a smart contract function "requestServiceAccess" that allows authorized service consumers to access an IoT device by paying a specified amount of service fee. The function starts by getting the address of the user who called the function and

```
function requestServiceAccess(address _iotDeviceAddress) onlyhasDeposited    infinite gas
 onlyAuthorizedServiceUser external returns (bytes32) {

address _userAddress = msg.sender;
uint256 _amountDeposited=userDeposits[_userAddress];

emit userhas(_amountDeposited);
emit cost(serviceTarrif[_iotDeviceAddress]);
// Check if the service tariff is less than the amount deposited
require(serviceTarrif[_iotDeviceAddress] <= _amountDeposited, "Not Enough Funds Deposited");

bool userAllowed = false;

string memory userRole = userAddressToRoleMapping[_userAddress];

for (uint i = 0; i < rolesAllowed[_iotDeviceAddress].length; i++) {
    string memory role = rolesAllowed[_iotDeviceAddress][i];
    if (keccak256(abi.encodePacked(role)) == keccak256(abi.encodePacked(userRole))) {
        userAllowed = true;
        break;
    }
}
// Check if the user is allowed to access the IoT device
require(userAllowed == true, "Illegal User");

address payable IotOwnerAddress = serviceToOwnerMapping[_iotDeviceAddress];

// Transfer the amount to the IoT device owner's address
IotOwnerAddress.transfer(_amountDeposited);

// Deduct the amount from the user's deposited amount
userDeposits[_userAddress] -= _amountDeposited;

bytes32 token = keccak256(abi.encodePacked(tokenNonce, _userAddress, _iotDeviceAddress));

tokenNonce++;

emit TokenEvent(_userAddress, _iotDeviceAddress, token);
tokensIssued[_userAddress].push(token);
return token;
}
```

Fig. 2 Example smart contract function

the amount deposited by the user. It then checks if the user has enough funds to pay for the service and if the user has the required role to access the device. If both conditions are met, the function transfers the payment to the owner of the IoT device and issues a token to the consumer to access the device.

Our smart contract, built on the Ethereum ecosystem, provides end-users and IoT owners with access via clients[26]. It facilitates access to IoT device data through a public platform, with payment details and steps outlined on the platform. The smart contract generates authorization tokens that users can use to authenticate themselves and access data off-chain. It offers various functions such as:

- userRegistration(): This function is executed by the role issuing authority to register and provide the requested role to the consumers.

- serviceProviderRegistration(): This function is used by the role issuing authority to register and provide role to the service provider.
- registerService(): This function is used by the service providers to add their IoT devices with the service platform.
- requestServiceAccess(): Service consumers execute this function to get the access for the IoT services.
- servicePayment():Service consumers execute this function to send payment required for accessing the IoT Device.

E. Decentralzied Service Discovery

In our IoT service platform, the metadata about the services is stored on the P2P network rather than the blockchain. The peers in the P2P network are nodes that form the blockchain network. However, the metadata about the services is not stored on the blockchain. This decision was made because the P2P network allows for more flexible and complex queries than the blockchain, which is better suited for recording immutable and tamper-proof transactions.

To enable efficient and flexible service discovery, we propose a two-layer query routing scheme. The first layer is a coarse-grained abstract search, which enables quick location of services in several main categories, while the second layer is a fine-grained detailed semantic search, which provides a semantically rich way of search of the services.

When service providers register their services to the blockchain, they also publish the metadata on the P2P network using a two-layer service publishing scheme. The first layer of service publishing is known as the service abstract publishing. It involves publishing the service abstract, which allows for the quick location of services in main categories like service type, location, provider, and price (as shown in Fig. 5's Service abstract example). These categories are published as key-value pairs and are propagated to all peers in the network, ensuring fast locating of services falling under these categories. For instance, a user can locate temperature sensors or sensors located in a particular location through this layer. The second layer, known as service description publishing, enables flexible and expressive querying by publishing service details using the ontology description. The ontology description provides a standardized and semantically rich way of describing services and includes essential metadata such as service name, type, Ethereum address, description, price, and more. For further details, refer to Section III.B.

Corresponding to the two service publishing schemes, two service query interfaces are provided. The first is category-based coarse grain query, which allows users to browse through the services and narrow down their search by specific categories, like finding temperature sensing services in a particular location. The second layer is a detailed semantic-rich query, such as a SPARQL query, which is matched at the abstract level first and then refined based on the results found in the first level. For example, a user is interested in finding temperature sensors in a particular location that provide data in a specific format, such as Fahrenheit. They could use a SPARQL query to search for this information using the query example listed in Section III.B.

IV. Evaluation

To assess the effectiveness of our proposed mechanism, we deployed it on an Ethereum blockchain network. This platform provides a higher level of security and privacy protection through the use of smart contracts. We conducted a comprehensive evaluation by performing a use case study that illustrates how smart contracts can be used to enhance security and protect IoT devices through user and device authentication, access control for IoT services, and secure payment. We then analyzed how our proposed mechanism protects IoT systems from various types of attacks and ensures security. Additionally, we showcased the robust search mechanism that enables discovery of IoT services through various search techniques. Finally, we assessed the performance of our decentralized service metadata indexing system, focusing on key metrics such as scalability, load balancing, and fault tolerance, to ensure optimal system performance.

A. Use Case Evaluation

Use case evaluations are a common method for assessing the functionality and effectiveness of a proposed system in a practical context. By using examples and specific scenarios, use case evaluations provide a more tangible and understandable way to assess the platform's capabilities and limitations. In essence, use cases allow us to demonstrate how the proposed platform addresses the challenges and limitations of current IoT service platforms and how it can provide secure and efficient services in practice.

- **Secure Access and payment**

This use case outlines how a service provider can securely connect their IoT device to our platform, allowing for automated usage by various consumers and generating revenue. For example, a device owner with a long-range temperature sensor can securely connect it to our platform, enabling automatic use and secure payments for their sensor.

To ensure security in the Ethereum ecosystem, both the service provider and consumer require public and private keys. In this use case, the IoT device owner (service provider) has the Ethereum Address 0xAb8…35cb2, while the service consumer has the Ethereum Address 0x4B2…C02db. The service provider registers their service: a temperature sensor, sets a tariff of $10 per day (5600000000000000 wei), and associates it with our platform. The consumer can use the semantic search on our platform to locate the required service provider with the address 0x787…cabaB. Security is maintained through the use of public and private keys. Fig. 3 illustrates the smart contract output when the role-issuing authority executes the register service function and provides the role "NDSUProvider" to the service provider. The output includes information about the unique hash of the transaction, transaction status, sender address details, function executed by the transaction sender, and logs generated by the function's execution, which includes event logs.

Fig. 3. Service provider is registered by the role issuing authority.

The service consumer (Ethereum address: 0x4B2…C02db) wishes to access the temperature service provided by the IoT device owner. The consumer checks the platform to obtain the service tariff and deposits the corresponding amount with the smart contract to gain access. In Fig. 4, the "payment for service access" function is executed by the consumer, allowing them to make a payment equal to the tariff amount to the smart contract.

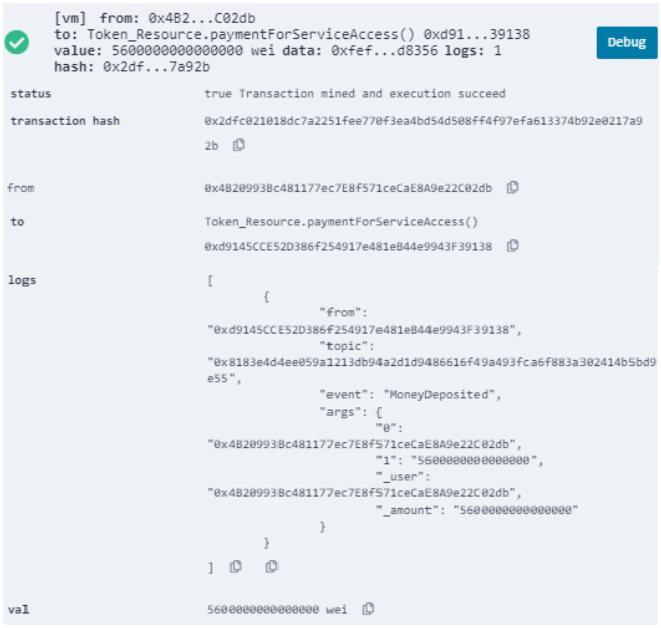

Fig. 4. Payment deposited by the service consumer.

Once the necessary amount is deposited, the consumer can use the "requestAccess" function on the smart contract to receive a token granting them access to the temperature service. The results of the smart contract are recorded in the logs, as shown in Fig. 5, which demonstrate the creation of the token that authorizes the service consumers to access the service.

Fig. 5. Service access token is generated for the consumer.

Fig.6 demonstrates that if the service consumer attempts to deposit an amount that is below the tariff set by the service provider and requests a service access token, the smart contract will automatically reject the transaction.

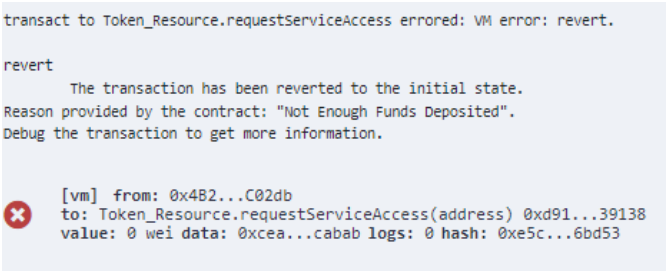

Fig. 6. The smart contract rejects service access requests that involve insufficient payment for the requested service.

In Fig. 7, it is demonstrated that if a different service consumer, with the Ethereum Address (0x17...8c372), attempts to access the temperature sensor service without providing the necessary tariff payment, the smart contract will automatically reject the transaction.

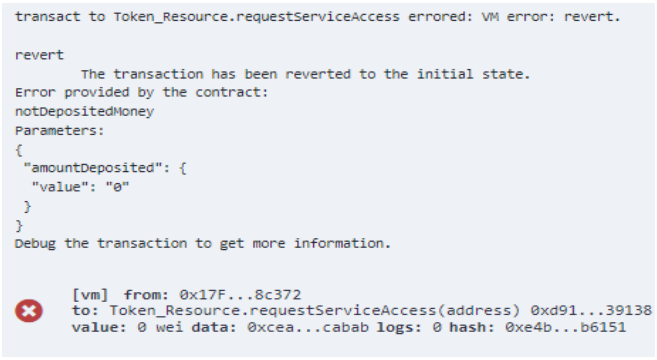

Fig. 7. Automatic rejection from smart contract on access request without payment.

Fig. 8. Automatic rejection from smart contract on access request for unregistered consumers.

Fig. 8 demonstrates that if an unauthorized service consumer, with the Ethereum Address (0x5c6…21678), attempts to obtain a token for service access without being registered with the platform through the role issuing authority, the smart contract will automatically reject the transaction.

B. Simulation Evaluation

Besides the security performance, we also designed a sequential network simulator to evaluate the performance of the proposed IoT network architecture. The simulator was implemented using a discrete event simulation approach, in which the simulation time is divided into discrete time steps. At each time step, the simulator updates the state of the network based on the events that have occurred since the previous time step. Events include things like a node joining the network, a node leaving the network, or a node requesting service from the IoT network.

The experiments evaluated the scalability of the proposed architecture. We used service request latency and network overhead as our evaluation metrics. Fig. 9 shows the network latency with the increasing number of users. Specifically, it compares three different system architecture: an ideal centralized cloud that has unlimited computing power, storage and bandwidth, a single server cloud with limited computing, storage, and bandwidth, and our decentralized blockchain-based system.

Latency is defined as the average time taken for an IoT service request to be responded. This can be expressed as:

$$D = Dprop + Dtrans + Dqueue + Dproc \quad (1)$$

In Equation 1, we can see that the delay of a request denoted as D, is dependent on several factors. Firstly, the propagation delay, *Dprop*, is the time taken for a signal to travel from the sender to the receiver, and is determined by the distance between the two parties and the transmission speed. Next, the transmission delay, *Dtrans*, is the time needed to transmit all data in a packet through the transmission link, and is determined by the packet length and transmission rate. The queuing delay, *Dqueue*, is the waiting time for a packet in a router's buffer before processing, which is influenced by the rate of incoming data, the outgoing link bandwidth, and network traffic. Lastly, the processing delay, *Dproc*, is the time for to process the packet and is dependent on the processing speed.

As shown in Fig. 9, the optimal cloud would result in the lowest latency, but this assumes an unrealistic scenario where the cloud has unlimited bandwidth and computing power. In contrast, a cloud with limited resources can result in significant latency or even become inoperable as the number of users increases, in our case at around 400 users. In contrast, our proposed decentralized system (with 500 P2P nodes) demonstrates linearly increasing latency in proportion to the number of users, which highlights the system's scalability in terms of service request latency.

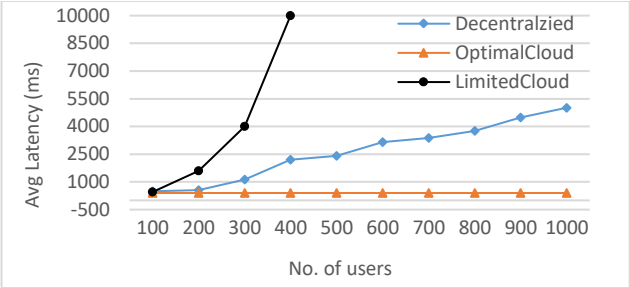

Fig. 9. Latency comparison of different network architectures with increasing user load.

In addition to evaluating the scalability of the proposed decentralized architecture in terms of service request latency, we also analyzed its network overhead. As shown in Fig. 10, we compared two networks with sizes of 500 nodes and 100 nodes, respectively. The results indicate that the network overhead (per node) increases in a linear fashion with the number of users. This finding reinforces the system's scalability, indicating that it can handle a growing number of users without significantly impacting its network performance. Overall, these results support the effectiveness of the proposed decentralized architecture in achieving both scalability and efficiency in the face of increasing user demand.

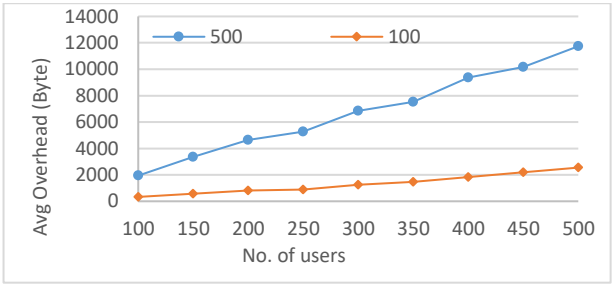

Fig. 10. Network overhead of different network sizes with increasing user load

V. Conclusion

This paper discusses the challenges faced by current IoT service platforms due to the increasing number of IoT devices. While there are numerous platforms available, they are typically designed for single domains, making it difficult to extend and integrate IoT services. To overcome these challenges, the paper presents an innovative solution - a flexible and open IoT service platform that enables easy integration and extension of services for multiple domains. The platform offers secure registration, discovery, access, and payment services to both providers and consumers, employing ontology-based descriptions and smart contracts for encryption, access control, and secure transactions. The platform also uses a decentralized peer-to-peer network for service registration and discovery, making it scalable and robust. The paper presents comprehensive evaluations through use case studies, and simulation experiments, demonstrating the platform's effectiveness. Overall, the proposed platform provides an open, extensible, and secure ecosystem that easily integrates IoT devices into new business and applications.

REFERENCES

[1] B. Pourghebleh, V. Hayyolalam, and A. Aghaei Anvigh, "Service discovery in the Internet of Things: review of current trends and research challenges," Wirel. Networks, vol. 26, no. 7, pp. 5371–5391, 2020, doi: 10.1007/s11276-020-02405-0.

[2] R. Basatneh, B. Najafi, and D. G. Armstrong, "Health Sensors, Smart Home Devices, and the Internet of Medical Things: An Opportunity for Dramatic Improvement in Care for the Lower Extremity Complications of Diabetes," J. Diabetes Sci. Technol., vol. 12, no. 3, pp. 577–586, 2018, doi: 10.1177/1932296818768618.

[3] L. Babun, K. Denney, Z. B. Celik, P. McDaniel, and A. S. Uluagac, "A survey on IoT platforms: Communication, security, and privacy perspectives," Comput. Networks, vol. 192, no. January, p. 108040, 2021, doi: 10.1016/j.comnet.2021.108040.

[4] G. Fortino, A. Guerrieri, P. Pace, C. Savaglio, and G. Spezzano, "IoT Platforms and Security: An Analysis of the Leading Industrial/Commercial Solutions," Sensors, vol. 22, no. 6, pp. 1–17, 2022, doi: 10.3390/s22062196.

[5] D. Wu et al., "A fog computing-based framework for process monitoring and prognosis in cyber-manufacturing," J. Manuf. Syst., vol. 43, no. 2017, pp. 25–34, 2017, doi: 10.1016/j.jmsy.2017.02.011.

[6] T. Wang, G. Zhang, A. Liu, M. Z. A. Bhuiyan, and Q. Jin, "A secure IoT service architecture with an efficient balance dynamics based on cloud and edge computing," IEEE Internet Things J., vol. 6, no. 3, pp. 4831–4843, 2019, doi: 10.1109/JIOT.2018.2870288.

[7] R. Sikarwar, P. Yadav, and A. Dubey, "A survey on IOT enabled cloud platforms," Proc. - 2020 IEEE 9th Int. Conf. Commun. Syst. Netw. Technol. CSNT 2020, pp. 120–124, 2020, doi: 10.1109/CSNT48778.2020.9115735.

[8] D. Pizzolli et al., "Cloud4IoT: A heterogeneous, distributed and autonomic cloud platform for the IoT," Proc. Int. Conf. Cloud Comput. Technol. Sci. CloudCom, vol. 0, pp. 476–479, 2016, doi: 10.1109/CloudCom.2016.0082.

[9] A. Taherkordi and F. Eliassen, "Scalable modeling of cloud-based IoT services for smart cities," 2016 IEEE Int. Conf. Pervasive Comput. Commun. Work. PerCom Work. 2016, 2016, doi: 10.1109/PERCOMW.2016.7457098.

[10] X. Li, Q. Wang, X. Lan, X. Chen, N. Zhang, and D. Chen, "Enhancing cloud-based IoT security through trustworthy cloud service: An integration of security and reputation approach," IEEE Access, vol. 7, pp. 9368–9383, 2019, doi: 10.1109/ACCESS.2018.2890432.

[11] S. S. Arslan, R. Jurdak, J. Jelitto, and B. Krishnamachari, "Advancements in distributed ledger technology for Internet of Things," Internet of Things (Netherlands), vol. 9, p. 100114, 2020, doi: 10.1016/j.iot.2019.100114.

[12] Q. Zhu, S. W. Loke, R. Trujillo-Rasua, F. Jiang, and Y. Xiang, "Applications of distributed ledger technologies to the internet of things: A survey," ACM Comput. Surv., vol. 52, no. 6, 2019, doi: 10.1145/3359982.

[13] B. Farahani, F. Firouzi, and M. Luecking, "The convergence of IoT and distributed ledger technologies (DLT): Opportunities, challenges, and solutions," J. Netw. Comput. Appl., vol. 177, no. September 2020, p. 102936, 2021, doi: 10.1016/j.jnca.2020.102936.

[14] S. Su et al., "IoT root union: A decentralized name resolving system for IoT based on blockchain," Inf. Process. Manag., vol. 58, no. 3, p. 102553, 2021, doi: 10.1016/j.ipm.2021.102553.

[15] L. Cui, S. Yang, Z. Chen, Y. Pan, Z. Ming, and M. Xu, "A Decentralized and Trusted Edge Computing Platform for Internet of Things," IEEE Internet Things J., vol. 7, no. 5, pp. 3910–3922, 2020, doi: 10.1109/JIOT.2019.2951619.

[16] I. Szilagyi and P. Wira, "Ontologies and semantic Web for the internet of things - A survey," IECON Proc. (Industrial Electron. Conf., pp. 6949–6954, 2016, doi: 10.1109/IECON.2016.7793744.

[17] D. Andročec, M. Novak, and D. Oreški, "Using semantic web for internet of things interoperability: A systematic review," Int. J. Semant. Web Inf. Syst., vol. 14, no. 4, pp. 147–171, 2018, doi: 10.4018/IJSWIS.2018100108.

[18] F. Z. Amara, M. Hemam, M. Djezzar, and M. Maimor, "Semantic Web and Internet of Things: Challenges, Applications and Perspectives," J. ICT Stand., vol. 10, no. 2, pp. 261–292, 2022, doi: 10.13052/jicts2245-800X.1029.

[19] S. Staab and R. Studer, Handbook on ontologies. Springer Science \& Business Media, 2010.

[20] M. Bermudez-Edo, T. Elsaleh, P. Barnaghi, and K. Taylor, "IoT-Lite: a lightweight semantic model for the internet of things and its use with dynamic semantics," Pers. Ubiquitous Comput., vol. 21, no. 3, pp. 475–487, 2017, doi: 10.1007/s00779-017-1010-8.

[21] M. Achir, A. Abdelli, L. Mokdad, and J. Benothman, "Service discovery and selection in IoT: A survey and a taxonomy," J. Netw. Comput. Appl., vol. 200, no. January, p. 103331, 2022, doi: 10.1016/j.jnca.2021.103331.

[22] S. Alian, J. Li, and V. Pandey, "A personalized recommendation system to support diabetes self-management for American Indians," IEEE Access, vol. 6, pp. 73041–73051, 2018.

[23] V. Pandey, J. Li, and S. Alian, "Evaluation and Evolution of NAOnto - An Ontology for Personalized Diabetes Management for Native Americans," 2021 7th Int. Conf. Comput. Commun. ICCC 2021, pp. 1635–1641, 2021, doi: 10.1109/ICCC54389.2021.9674339.

[24] Vitalik Buterin, "pos_faq @ vitalik.ca." [Online]. Available: https://vitalik.ca/general/2017/12/31/pos_faq.html.

[25] S. Wang, Y. Yuan, X. Wang, J. Li, R. Qin, and F. Y. Wang, "An Overview of Smart Contract: Architecture, Applications, and Future Trends," IEEE Intell. Veh. Symp. Proc., vol. 2018-June, no. Iv, pp. 108–113, 2018, doi: 10.1109/IVS.2018.8500488.

[26] Buterin and Vitalik, "Ethereum White Paper: A Next Generation Smart Contract & Decentralized Application Platform," Etherum, no. January, pp. 1–36, 2014, [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper