An autonomous loyalty program based on blockchains for IoT solution providers

Shahin Gheitanchi

IEEE Senior member

Gheitanchi@ieee.org

2020 IEEE Global Conference on Artificial Intelligence and Internet of Things (GCAIoT) | 978-1-7281-8420-3/20/$31.00 ©2020 IEEE | DOI: 10.1109/GCAIOT51063.2020.9345892

***Abstract*— Data exchange and sharing methods for internet of things are widely studied and implemented, mostly in isolation from business considerations. In this paper, we introduce the concept of an autonomous loyalty program for IoT operators and explore its implementation. The purpose of the concept is to establish connection between business and technology layers by design, and in an autonomous fashion. The proposed concept utilizes tokenized economy on blockchain where the operators can implement gamification techniques using smart contracts to maximize profit during service offerings and requests. The system model, exchange procedures and implementation of the concept are discussed.**

Keywords—IoT, Blockchain, Gamification, Service Exchange

Managerial impact: The proposed concept of loyalty program would enable IoT solution providers and network operators to implement and automate gamification at the device level. This could generate financial incentives for solution providers to share and also to use shared resources, and hence reduces costs and time-to-market. The proposed method utilizes blockchain technology which enables secure, transparent and peer-to-peer dealing without 3rd party involvement.

I. Introduction

Internet of things (IoT), has positively impacted many businesses. This includes telco operators which profit from deployment and selling services of the IoT devices. However, it is important for the operators to quickly reposition their business and technology settings according to market dynamics [1].

Utilizing shared and ready-to-use resources, when its profitable to do so, is a well-known method to address market dynamics in a short period of time and de-risk development challenges. For example, a sensor which is located in a certain place could be shared among IoT network operators. There have been efforts to enable resource sharing mostly by using centralized cloud services [2] [3] [4] [5]. Also, more recently, decentralized architectures and peer-to-peer exchanges [6] [7] [8] have been considered where the focus have mainly been to address trust and privacy concerns of centralized methods. Further to the proposals for connectivity and exchange mechanisms, in [9], the concept of adding gamification techniques [10] to a service exchange platform as way to achieve business goals has been proposed.

In this paper, we further elaborate on the idea of gamification and explore its implementation in context of loyalty program for IoT service providers in order to minimize the gap between technology and business layers. We use blockchain technology [11] to facilitate trust, transparency and transfer of value which is essential for autonomous peer-to-peer service exchange. To remain within the scope of the paper, technical implementation details have been intentionally limited to high level description and mentioning of practical consideration. In the rest of paper, we provide a brief overview of current gamification application in IoT. Next, the loyalty program architecture based on the concept from [10] is reviewed. In section IV, we describe structures and procedures for service offerings and requests on for the loyalty program. In section V, a high-level description of the concept implementation is presented. In section VI, implementation considerations are described.

II. Gamification for Iot

For many years, gamification techniques have been used to increase user engagement by motivating people to achieve their personal and organizational goals. The techniques have been used for marketing, education, healthcare and motivating employees [10]. This is done by defining individual players goals considering big picture goals and designing game plays and mechanics to achieve the goals. Game plays are journey plans and stories for the users to follow and the game mechanics are the rules and interaction methods. Gamification techniques award activities that are in accordance with predefined goals and remain neutral or discourage the rest.

Moreover, gamification techniques have been widely applied to maximize user engagement with IoT devices in various fields such as energy conservation [12], healthcare [13], and safety [14]. The techniques are only applied at the application layer where humans interact with IoT solutions. The purpose is often to emotionally engage and motivate the end users to interact with the devices. The infrastructure of an IoT solution, such as cloud servers, gateways and devices, which provide the required services for the application layer is not included in the gamification. The main reasons are lack of human emotions as a decision-biasing factor and also absence of an enabler platform to run game plays and mechanics in the IoT infrastructure. With the current trend of technology where more intelligence is being built into devices [15] [16] [17] [18] more sophisticated and autonomous behavior from IoT infrastructures could be expected. The intelligence could be utilized for creating decision-bias toward an action where it maximizes the benefits for the solution provider business.

As a gamification enabler platform, centralized and decentralized methods could be used. However, for true autonomous peer-to-peer exchange, blockchain technology is a suitable candidate. Recently, blockchain technology has been considered as gamification enabler platform due to its transparency and tokenized features [19].

III. Loyalty program architecture

A. System overview

A generalized IoT network consists of autonomous, heterogenous and geographically dispersed entities where each entity has two main layers: technology and business. The technology layer, also called IoT stack, consists of devices, gateways, cloud and application. In our proposed system which is based on [9], the entities are connected to blockchain using oracles which are light software modules that act as a bridge between blockchain and other layers of an entity. Oracles invoke smart contacts, transfer tokens using wallets, and decide to utilize internal or external resources based on the parameters obtained from business layer. The business layer provides the oracle with pricing, reward for gamification, legal requirements and duration of service. Also, the business layer provides internal cost of a service and preference parameters for using internal or external resources to the oracle during service requests. Devices, gateways, cloud servers and application layers are assumed to have predefined atomic services which can either be imported or exported with terms defined by the business layer. The IoT network operators independently and individually can govern policies in the business layer which makes the entities autonomous.

Blockchain technology [11] is utilized as the facilitator of peer-to-peer exchange of service and value. It is based on decentralized servers, called nodes, that hold a copy of transactions and service states in form of cryptographically connected data blocks known as the distributed ledger. The nodes remain in sync via consensus algorithms which determine how the nodes are updated and synchronized. Moreover, tokens are defined and utilized for value transfer in the blockchain. The nodes act as a virtual machine and can run software codes, also called smart contracts, which can automatically alter the states of a smart contract and transfer tokens. The proposed loyalty program architecture is independent of the blockchain infrastructure and can be implement on any blockchain that supports smart contracts.

B. Gamification modules

The purpose of the gamification modules is to provide the necessary tools for implementing gamification strategies to maximize utilization of the services offered by an entity and hence increase its financial gains. The gamification modules implement the following two categories of functionalities:

- Game mechanics: Includes tokens, wallets and smart contracts. Tokens are used as a mean to transfer value and reward according to the game story. Wallets are essential parts of game mechanics to store and transfer tokens. Smart contracts on the blockchain facilitate transfer of tokens among entities. Game mechanics are implemented partly on entities' oracles and partly on blockchain network.
- Game story: It is a set of policies and configurations to bias service requesting entities to repeatedly use the offered services by making service outsourcing cost efficient. It includes the list of status, conditions to reach the status and the rewards given away to the entities and is implemented in the business layer.

Figure 1 demonstrates only the entity stack layers which implement the gamification modules. The dotted arrows illustrate logical connection among business layers of participating entities where the game story is applied. For offering a service, the business layer of the entity adjusts service offering price and reward according to game story and behavior of other entities to maximize service utilization and hence its financial profits. When an entity requests a service, the business layer adjusts the preference parameter and provides it to entity's oracle for decision making. Further detail on service offers (SO) and service requests (SR) is described in the following section.

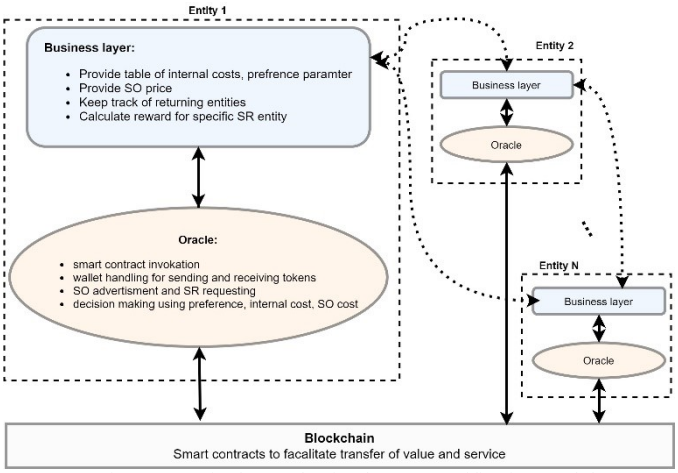

Figure 1: Entity layers that implement gamification modules.

Unique entity identifiers (IDs) are used by business layers to keep track of previous service usage of an entity and decide on the entity's status and reward. The number of reward tokens can be affected by time and load of the offering entity.

IV. Service offers and requests

Each entity can offer and also request services. A service can include real-time data, offline datasets, and functionalities which are defined and offered by one or more layers of the IoT stack in an entity. The offered and requested services can be pre-defined at stack design time or adaptively occur during entity's operation. The services are packaged in an atomic form where they can start and stop on given terms without any dependencies before or after. Service offering and requesting procedures are initiated by IoT stack in an entity to export or import a service, respectively.

A. Structures

Service offers and requests have similar structure fields which are populated with information provided by business

layer and, one or more layers of the IoT stack. Oracles are responsible to encapsulate the provided information in service offer (SO) and service request (SR) structures that could be implemented using attribute-value data structures such as JSON. Table 1 describes the service information fields of SOs and SRs. Some fields such price and reward are unused in SRs.

SRs and SOs have one-to-many relation where a SR can utilize a single SO while a SO may be used by many SRs from multiple service providers. Therefore, SOs are designed in a way to be general and modular so that can match a wide range of SRs to maximize financial profits of the offering entities.

TABLE I. SO AND SR INFORMATION FIELDS.

Fields	SO	SR	Purpose
Service type	✔	✔	It describes type of service and list of offered or requested data (i.e. temperature data) and functionalities (i.e. historical data processing).
Duration	✔	✔	It includes duration of service that is offered or requested (i.e. 24 hours).
Delivery	✔	✔	Indicates the communication protocol, format of data and method of online or offline service delivery.
Legal	✔	✔	Provides data for regulatory and legal purposes such as consents and geographical restriction on usage of data (i.e.: EU countries only).
Price	✔	✘	The number of tokens needed for the service (i.e. 500 tokens).
Reward	✔	✘	Optional number of tokens given back as reward which are used by gamification policies in business layer (i.e. 50 tokens).

B. *Procedures*

Offering a service is about making service information which is given in Table 1 available to other entities and it is a two-stage process. At the first stage, the oracle advertises the service on blockchain using a SO structure where the other entities can find it. The SOs can be advertised on the blockchain using a marketplace smart contract that holds the references to SOs or on a separate off chain cloud that solely acts as a marketplace and redirects the rest of operations to the blockchain. Using off chain marketplace has the benefit of utilizing optimizations that apply to centralized systems. The choice of SO advertisement implementation depends on underlying blockchain technology limitations such as transaction costs and processing durations. At the second stage, the ID of service requesting entity is received by the oracle of the offering entity and is passed to the business layer where the gamification policies are applied. Depending on the previous service usage of the requesting entity, the status and the relevant number of reward tokens are decided.

Requesting a service is defined as finding the cheapest way of addressing IoT stack requirements by comparing cost of available SOs and cost of using internal resources. It includes discovery, handshake, and decision-making steps. In the discovery step, the IoT stack layers call an application programming interfaces (APIs) in oracle which takes services information as its arguments, creates a SR, and explores the blockchain for matching SOs. The API matches service information except the reward field that will been obtained during handshake step.

The discovered SOs are returned to the requesting oracle in form of an array. In the handshake process, ID of the service requesting entity will be broadcasted to the matching SO entities where the ID gets processed by the business layer to obtain the rewards for the specific SR entity. In the decision-making step, the SR entity will check the price and reward fields of the SOs and use the following optimization function to decide whether to use internal resources or the SO with minimum cost. If the cost of addressing a SR externally by a SO is less than the cost of utilizing internal resources, then the smart contracts on the exchange platform are used to automatically forge a contract and start online or offline service delivery.

V. IMPLEMENTATION OVERVIEW

In this section, overview of an example implementation of the proposed architecture is explained in a high level to demonstrate practicality and scalability of the concept. To keep the section within the scope of the paper, only the technologies, gamification module methods and an example scenario are briefly presented. All the IDs and addresses used in the example scenario are presented as integer numbers instead of real unique universal IDs to improve readability. Throughout the section, we consider a pool of solution providers which are connected to a distributed virtual machine (DVM) via oracles. The block diagram of the concept implementation is shown in Figure 2.

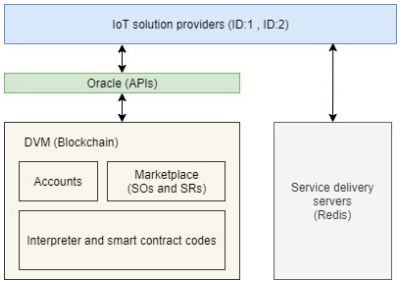

Figure 2: Block diagram of concept implementation.

A. *Solution providers*

To better explain the implementation, an example scenario is used where a solution provider (ID: 1) requires temperature readings from various locations of a city for its new product. There is another solution provider (ID: 2) which already has the equipment rolled out in the city and can provide the readings. The solution providers are simulated by NodeJS instances. Each solution provider has a business layer similar to the one explained in section III-A. It is assumed that the business layers of solution providers have knowledge of their own legal limitations, internal costs for delivering the service and gamification rules. Example of the business layer of an offering solution provider is:

```
SOArray= [ {SOID: "152", type: "temperature", duration: [1, "d"],
dID: 1, legal: {location: ["EU"]}, price: 10, pID: 1 }]
DeliveryMethods = [{platform: "Redis", dataType: "JSON"}]
GamePlays = [{count: [2, 5, 10] , discountPercent:[5,10,15] }]
PastClients = [{cID:11, count: 5},{cID:14, count: 1}]
```

where *SOArray* is the array of the solution provider's active SOs. *DeliveryMethods* is the array of supported delivery methods by the solution provider and includes delivery platforms and data types. *GamePlays* encompasses various game plays. In the above example, the only game play is to offer 5, 10 and 15 percent discounts to the number of reuses of 2, 5 and 10 times, respectively. *PastClients* is an array that holds history of past client IDs and their usage counts. The history along with game plays are used to determine the reward. Each SO in *SOArray*, holds the index of delivery methods (*dID*) and the index of game play (*pID*).

The requesting entity's business layer stores the internal costs and the counterpart of the above-mentioned items as following

```
iCost = [{type: "temperature", duration:[1,'d'], cost:30, w:0.5}]
SRArray = [{SRID:"21", type:"temperature", duration:[1,"w"],
           dID:1, legal:{Location:["EU", "US"]} }]
DeliveryMethods = [{platform: "Redis", dataType: "JSON"}]
```

where *iCost* is the array of the internal cost of a service type in term of tokens for a given duration and also preference weight. *SRArray* and *DeiliveryMethods* hold the active SRs and supported delivery methods, respectively. The above information is utilized for advertising SOs and SRs on marketplace, handshake, decision making as described in following sections.

B. Oracles

In this layer, the DVM low-level instructions are encapsulated in form of APIs and are accessible via each blockchain node. Tables II shows oracle introduction sets for the exchange platform operations.

TABLE II. ORACLE API COMMANDS.

Instruction	Purpose	Params
addSO	Add a service offering	JSON object: [ID, type, legal, duration, delivery, price, reward]
rmSO	Remove a service offering	String: SO ID
addSR	Add a service request	JSON object: [ID, type, legal, duration, delivery mechanism]
rmSR	Remove a service request	String: SR ID
txMsg	Message transaction between service offering and requesting entities	JSON object: [from, to, message, value1, value2]

The *txMsg*, described in Table II, is a multiple purpose command which allows its functionality as described in Table III. Furthermore, the decision-making process will occur in this layer which is elaborated more in the following section C-3.

TABLE III. TXMSG COMMAD PARAMTERS.

Purpose	Message	Value 1	Value 2
Request price and reward for an SR ID	handshake	ID	Null
Handshake acknowledge	handshakeAck	Price	Reward
Requesting service	serviceReq	SO_ID	Tokens
Acknowledge service request	serviceReqAck	Delivery detail	Null

C. Decentralized virtual machine

It is a decentralized computer where the nodes can join and disjoin at any time. It maintains a global state of the system that includes all key-value pairs that can be altered via transactions or a computer program. Each node keeps a record of current state and history of transaction, and it is able to execute instructions. The consensus method to synchronize the nodes is proof of work [20] which ensures the transactions are mined in a timely manner by adaptively controlling complexity of mining algorithms. The SO or SR related procedures do not require deterministic and low latency transactions, while the service delivery may have restriction on timing requirements. Therefore, the service delivery part, explained later in this section, is designed to be out of the blockchain and be a stand-alone network. The underlying data structure for the DVM is a blockchain with ability to execute instructions, which is implemented using JavaScripts and is inspired by Ethereum [20]. The reason for not using existing public blockchains is to ensure sufficient throughput and manageable costs. DVM also includes low-level instructions and an interpreter to execute them for creating smart contracts. Examples of low-level instructions are *ADD* for addition of numbers, *JUMPI* for conditional jump and loop creations, *LOAD* to load data from smart contract account storage. These instructions are encapsulated in oracle instructions given in Table II. Furthermore, DVM holds track of two types of accounts using array structures. One type of account is associated with each solution provider where it holds the number of its tokens. The other type stores tokens used for a set of instructions, also known as smart contact, running on DVM. The smart contracts can automatically, based on pre-defined condition, transfer tokens among solution provider accounts and their accounts. Using DVM includes a cost which is paid from the account that initiates a transaction. For simplicity, in the presented scenario we assume the cost is zero.

The marketplace is an event driven service provided as part of the DVM for the solutions providers. It's a software written in JavaScripts and its main purpose is to accommodate structures and procedures for advertising and discovery stages mentioned in section IV.

1) Advertising SOs and SRs:

Advertising SOs is performed by the offering entity where it populates the following JSON object using the information embedded in its business layer. It submits the information to marketplace using *addSO* API described in Table II. The following code list demonstrates an example SO:

```
{ID: "1", type: "temperature", duration: [1, "d"],
delivery: {platform:"Redis", dataType: "JSON"},
legal: {location: ["EU"]}, price: 10, reward: null}
```

where the service ID is 152, the service type is temperature and it is provided for duration of 1 day for initial price of 10 tokens. For legal compliance, the service offering is limited to EU. In this implementation for simplicity, the legal compliance is limited to location constrains, however, it can include other fields such as end-user types. The service delivery mechanism is Redis [21] with JSON data type. The service exchange service will be explained later in this section. During SO listing, the reward can have a value if the solution provider is willing

to provide a discount. Otherwise, it will be null till a matching SR requests the final price in handshake process.

Adding a SO to the marketplace will make it visible and allows discovery of the service. Furthermore, the marketplace will notify any standing SR entity in the marketplace which matches the service. The provider of the SO is responsible to maintain an up-to-date version of the SO on the marketplace by using *addSO* and *rmSO* APIs. On the other hand, a SR is listed on the marketplace using *addSR* API. The following JSON object shows an example service request that is submitted to the marketplace

{ID: "2", type: "temperature", duration: [1, "d"],
delivery: {platform: "Redis", dataType: "JSON"},
legal: {location: ["EU", "US"]} }

In this example, the entity ID is 2 and the requested service type is temperature for duration of 1 day where the data can be accepted on Redis platform with JSON format. The service can be accepted from EU and US locations.

2) Discovery and handshake

By submitting a SR or a SO, the marketplace automatically triggers a discovery operation using the flowchart in Figure 3 which is implemented using JavaScripts.

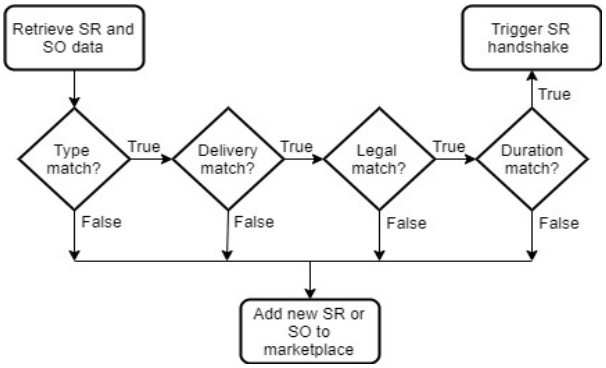

Figure 3: Service discovery flowchart.

The discovery operation for the presented marketplace scenario will results in a single element array SO = [ID: 1]. The next step is to handshake and find out the service cost based on the price and reward. The marketplace, automatically sends the ID of SR to the list of service providing entities. In return, the service providers use *GamePlays* and *PastClients* arrays in business layer to determine any possible reward. The handshake is performed using *txMsg* command, triggered by the marketplace smart contract. Assuming that the requesting service provider has previously used the service 3 times (count=3), according to counts and *discountPercent* of *GamePlays* array, the count falls between 2 and 5, and hence the 5% discount applies to the initial price. Therefore, the reward for price of 10 tokens is 0.5 tokens and the cost of service for requesting provider based on handshake result for the example scenario will be 9.5 tokens.

3) Decision making

After the handshake process, the requesting solution provider needs to make a decision for using internal or external resource by applying the optimization function described in section IV. This is done using business layer data in *iCost* array and C_{SO}. The cost of SO from handshake process is for duration of 1 day while the duration of SR is for 1 week. Therefore, the cost of SO for 7 days will be $C_{SO} = 7 \times 9.5 = 66.5$ tokens. The internal cost of the service from *iCost* array for duration 1 day is 30 tokens. Therefore, $C_{Internal} = 7 \times 30 = 210$ tokens where the preference weight, w, is 0.5 . Using a minimization function, the weighted internal cost, $w \times C_{Internal}$ is 105 tokens which is greater that the C_{SO}. Therefore, the decision-making process results in use of the minimum cost and hence utilizes the data provided by the SO solution provider rather than internal resources. At this stage the service requester will indicate willingness of using the SO by send a *serviceReq* message, using *txMsg*, to the service provider which transfers number of needed tokens and the relevant SO ID. The service provider, will acknowledge the token payments and provide connection details for the service exchange. The next step is to subscribe to the relevant channel on service exchange server and start the receiving the data.

D. Service exchange server

In order to support various delivery timing, scheduling, access permission and also provide a cost-efficient way of service delivery, a dedicated server next to DMV is designated for service exchange purpose only. The server is operated by the SO owner where each offered service can have a dedicated channel. Following a pubsub pattern [22], each service requesting entity can subscribe to SO channel where the credentials are provided at last stage of decision-making process. The server is implemented using Redis. The security and privacy can be improved by implementing encryption and authentication.

VI. Implementation Considerations

A. Business and Legal

Gamification concepts and business models for humans are well known [10]. Expanding gamification concepts to machines and providing an enabler platform, would open up a new branch in gamification studies to define new business models and methods to build intelligence to the devices to maximize their profits. Making the connection between business objectives and technical requirement of solution provider entities is essential to financially motivate the providers to share and reuse services.

In contrast to the centralized IoT solutions where the locality of solutions providers and servers are well known, in blockchain, the decentralized exchange servers could exist anywhere in the world. To commercialize the proposed architecture, jurisdictions laws may affect the deployment of blockchain nodes and services. Moreover, in the proposed architecture, the machines will autonomously agree on SO and SR terms, service exchange, and transfer tokens. Traceability and legality of the process needs to be clearly defined to stand in courts in case of dispute. Furthermore, the data on blockchain is immutable. This may be in violation of regulations such as "right to erasure" in EU [23].

B. Technical

Choice blockchain: The choice of public or private blockchain technology will affect performance, cost and security of the operations. Public blockchains such as Ethereum [24] will provide a wide range of development and integration tools, as well as making it easy to host a node for processing transactions. In return, they may have variant cost of usage and can put privacy and security at risk as the transactions data are available to the public. In contrast, a custom developed private blockchain can give more control on participating nodes and can run cheaper in long term. Getting easy access to tokens and converting them to fiat currencies will also depend to the selection of blockchain. The choice of blockchain technology also affects some IoT applications that may require deterministic latency. Transaction processing in public blockchains may introduce large jitter in service delivery time which may not be acceptable for some IoT solutions.

Security and privacy: To provide sufficient security, a mechanism to ensure authenticity of entities is needed. Certificate authorities, provisioning methods and digital signatures could be a used to increase the integrity and security. Moreover, to protect privacy and security of service deliveries including data sets and device readings, encryption mechanisms can be used.

Power efficiency: Most IoT devices aim to run on batteries on over a long period of time. Minimizing computational complexity of entities' business layers, oracles and communication intervals will help with saving energy of the devices.

VII. Conclusion

In this paper, the concept of an autonomous loyalty program for IoT solution providers has been introduced. The concept utilizes tokenization and decentralized features of blockchain technology to enable peer-to-peer exchange based on gamified decision making. The architecture and implementation of the proposed concept was explored in high level to demonstrate practicality and scalability. The autonomous loyalty program would allow the IoT solution providers to implement their business goals in technology layer and allow machines to discover the suitable resources to address their requirements in a short period of time.

References

[1] P. Krussel, Future Telco: Successful Positioning of Network Operators in the Digital Age, Springer, 2018.

[2] L. Atzori, A. Iera and G. Morabito, "The Internet of Things: A survey," *Elsevier Computer Networks,* vol. 54, no. 15, 2001.

[3] R. Girau, S. Martis and L. Atzori, "Lysis: A Platform for IoT Distributed Applications Over Socially Connected Objects," *IEEE Internet of Things Journal,* vol. 4 , no. 1, 2017.

[4] F. Li, M. Voegler, M. Claessens and S. Dustdar, "Efficient and Scalable IoT Service Delivery on Cloud," in *IEEE Sixth International Conference on Cloud Computing*, 2013.

[5] P. Grubitzsch, T. Springer, T. Hara, I. Braun and A. Schill, "A Concept for Interoperable IoT Intercloud Architectures," in *7th International Conference on Cloud Computing and Services Science*, 2017.

[6] K. Christidis and M. Devetsikiotis, "Blockchains and Smart Contracts for the Internet of Things," *IEEE Access,* vol. 4, 2016.

[7] W. Chen, M. Ma, Y. Ye, Z. Zheng and Y. Zhou, "IoT Service Based on JointCloud Blockchain: The Case Study of Smart Traveling," in *IEEE Symposium on Service-Oriented System Engineering (SOSE)*, 2018.

[8] Z. Huang, X. Su, Y. Zhang, C. Shi, H. Zhang and L. Xie, "A decentralized solution for IoT data trusted exchange based-on blockchain," in *3rd IEEE International Conference on Computer and Communications (ICCC)*, 2017.

[9] S. Gheitanchi, "Gamified service exchange platform on blockchain for IoT business agility," in *IEEE ICBC2020*, 2020.

[10] B. Burke, Gamify: How Gamification Motivates People to Do Extraordinary Things, Routledge, 2014.

[11] H. Diedrich, Ethereum: Blockchains, Digital Assets, Smart Contracts, Decentralized Autonomous Organizations, CreateSpace Independent Publishing Platform, 2016.

[12] T. G. Papaioannou and e. a. Dimos Kotsopoulos, "IoT-enabled gamification for energy conservation in public buildings," in *Global Internet of Things Summit (GIoTS)* , 2017.

[13] M. U. Ahmed, S. Begum and W. Raad, Internet of Things Technologies for healthcare, Springer, 2016.

[14] K. Bahadoor and P. Hosein, "Application for the Detection of Dangerous Driving and an Associated Gamification Framework," in *IEEE International Conference on future Internet of Things and Cloud Workshops (FiCloudW)*, 2016.

[15] S. Gheitanchi, F. Ali and E. Stipidis, "Trained particle swarm optimization for ad-hoc collaborative computing networks," in *AISB 2008 Convention Communication, Interaction and Social Intelligence*, 2008.

[16] T. Qiu, N. Chen, K. Li, M. Atiquzzaman and W. Zhao, "How Can Heterogeneous Internet of Things Build our Future: A Survey," *IEEE Communications Surveys & Tutorials ,* 2018.

[17] F. Jalali, O. J. Smith, T. Lynar and F. Suits, "Cognitive IoT Gateways: Automatic Task Sharing and Switching between Cloud and Edge/Fog Computing," in *Proceedings of the ACM SICOMM Posters and Demos*, 2017.

[18] D. Schatsky, N. Kumar and S. Bumb, "Intelligent IoT Bringing the power of AI to the Internet of Things," Deloitte, Dec 2017. [Online]. Available: https://www2.deloitte.com/insights/us/en/focus/signals-for-strategists/intelligent-iot-internet-of-things-artificial-intelligence.html. [Accessed May 2018].

[19] R. Jackson, "How Blockchain Technology Is Bringing Gamification to Every Industry," April 2018. [Online]. Available: https://www.nasdaq.com/article/how-blockchain-technology-is-bringing-gamification-to-every-industry-cm924006. [Accessed May 2018].

[20] "Ethereum Yellow Page," 2019. [Online]. Available: https://ethereum.github.io › yellowpaper › paper.

[21] "Redis," 2019. [Online]. Available: www.redis.io.

[22] "pubsub pattern," 2019. [Online]. Available: https://en.wikipedia.org/wiki/Publish-subscribe_pattern.

[23] "Right to erasure," 2018. [Online]. Available: https://gdpr-info.eu/art-17-gdpr/.

[24] [Online]. Available: www.ethereum.org.