A Blockchain-Based Model for Cloud Service Quality Monitoring

Mona Taghavi, Jamal Bentahar, Hadi Otrok, and Kaveh Bakhtiyari

Abstract—This paper introduces a novel blockchain-based decentralized federation model that embodies quality verification for cloud providers who lease computing resources from each other. The blockchain structure removes the barriers of a traditional centralized federation and offers a fully distributed and transparent administration by enforcing the involved agents to maintain consensus on the data. For a blockchain-based federation, it is vital to avoid blind-trust on the claimed SLA guarantees and monitor the quality of service which is highly desirable considering the multi-tenancy characteristic of cloud services. Due to the fact that the blockchain network is unable to access the outside world, it cannot handle, by its own, providers misbehavior in terms of SLA violations. Thus, we introduce oracle as a verifier agent to monitor the quality of the service and report to the smart contract agents deployed on the blockchain. Oracle is a trusted third-party agent who can communicate with the outside world of the blockchain network. The interaction between cloud service providers (either providing a service or requesting it from another provider) and the oracle through smart contracts comprises a system of autonomous and utility maximizer agents. Cloud requesters seek to receive high quality services with constant monitoring at cheap prices or even with no charge, while cloud providers aim to have a balanced work-load with less preserved capacity, and the oracle tends to charge higher for their monitoring services. Therefore, to model this conflicting situation, we formulate a dynamic Stackelberg differential game to optimize the cost of using the oracle and maximize the profit of the agents with the role of provider agent as a leader, and the requester and verifier agents as followers. Our designed Stackelberg differential game can seize the dynamicity of users' demand and resource provisioning in a competitive cloud market. We implemented our proposed decentralized model using the Solidity language in the remix IDE on the Ethereum network. We further evaluated the optimal controls and agents' profit with real-world data simulated for three concrete cloud providers. The results revealed that the requester agent initiates most of the quality verification requests at the beginning to the middle time of the contract. Thus, the provider agent could reserve less computing resources considering the fact that it could share the workload among other customers' computing resources during the peak-time. Moreover, imposing a higher penalty on the provider agent increased the capacity and decreased the number of requests for quality verification at the equilibrium. The evaluation also disclosed that the impact of timing in the dynamic pricing strategy of the verifier agent is very minimal, and the provisioning capacity of the provider is strongly correlated with the monitoring price.

Index Terms—Smart contracts, quality verification, stackleberg differential game, cloud computing, service provider, oracle

1 Introduction

The demand variation has forced cloud providers to preserve a massive amount of computing resources to avoid Service Level Agreements (SLA) violation. To mitigate the issue of underutilized and over provisioned computing resources, cloud providers scaled their pool of resources by forming cloud federations to maximize their profit and provide guaranteed Quality of Services (QoS) [1]. In spite of their prominent advantages, cloud providers are reluctant to participate in federations due to some strict challenges, including the federations' stability, long-term commitments from the providers, fair revenue sharing, the presence of unknown and untrusted participants, security and privacy concerns regarding the managed data, and the creation and management overhead of these federations [1], [2], [3], [4]. In order to overcome the aforementioned limitations of the traditional federations, Cloudchain [5] proposed a new distributed blockchain-based framework to support interoperability and coopetition (i.e., cooperative competition) among the cloud providers. Cloudchain allows the cloud providers to outsource their unmet computing demands and agree on the values of shared variables (e.g., amount of the resource, SLA and price) and keep a history of how the values change over time.

Utilizing smart contracts in blockchain enabled Cloudchain to offer higher transparency, visibility, and reliance within its fully decentralized agreements deployed on top of Ethereum. However, Cloudchain falls short in supervising the SLA's agreed terms, which requires to access the outside world of the blockchain network. Each of the cloud providers may disagree about SLA compliance. However, investigating that is beyond the control of blockchain miners or digital codes embedded in the smart contracts due to its self-contained execution environment. Thus, we need a third party to perform the highly important verification task and confirm if the SLA is met.

- *H. Otrok is with the Center for Cyber-Physical Systems (C2PS), Department of EECS, Khalifa University, Abu Dhabi, UAE, and also with the Concordia Institute for Information System Engineering, Concordia University, Montreal, QC H3G 1M8, Canada. E-mail: Hadi.Otrok@kustar.ac.ae.*
- *M. Taghavi and J. Bentahar are with the Concordia Institute for Information System Engineering, Concordia University, Montreal, QC H3G 1M8, Canada. E-mail: mona.taghavi@gmail.com, bentahar@ciise.concordia.ca.*
- *K. Bakhtiyari is with the Interactive Systems, University of Duisburg-Essen, Duisburg 47057, Germany, and also with the Department of Electrical & Electronic Engineering, Universiti Kebangsaan Malaysia, Bangi, Selangor 43600, Malaysia. E-mail: academic@bakhtiyari.com.*

Manuscript received 31 Mar. 2019; revised 3 Sept. 2019; accepted 10 Oct. 2019. Date of publication 17 Oct. 2019; date of current version 15 Apr. 2020.
(Corresponding author: Mona Taghavi.)
Digital Object Identifier no. 10.1109/TSC.2019.2948010

To address the quality monitoring issue, we propose to employ oracles tailed to smart contracts within an innovative multiagent decentralized model. A smart contract is a piece of code deployed and executed on blockchain. Oracle in the blockchain context is a fully-trusted third-party agent that has access to the outside world, and feeds the data into the blockchain to be accessible by the applications. Oracles usually provide proofs to show that the retrieved data is tamper-proof. A number of oracles have been deployed using cryptographic evidence (e.g., hash code) such as Oraclize, or the Intel SGX feature, such as Town Crier, to make sure the data is tamper-proof. Our proposed multiagent model includes five different agents, namely the cloud service requester (requester agent (RA)), cloud service provider (provider agent (PA)), oracle (verifier agent (VA)), and two smart contract agents called registry-profile and contract agents. These contracts that were developed in [5], are registered in the blockchain and are triggered by new transactions (i.e., initiating new requests or registering inputs from VA), which will make each blockchain node update its state based on the results obtained after running the smart contract. The smart contract is considered as an agent that has state variables and enforces the associated rules. In our scenario, RA makes a contract with PA and might initiate a quality monitoring request anytime from VA. VA can check the quality of the service with respect to different attributes (e.g., availability, bandwidth, response time, etc.) and detects any misbehavior of RA or PA, then returns the result to the contract agent to manage the payments and apply potential punishments. Accordingly, the contract agent decides who should pay the monitoring cost to VA.

Having cloud requesters, providers, and the oracle interacting with each other through smart contracts composes a system of autonomous and utility maximizer agents. Cloud requesters seek to receive high quality services with constant monitoring which could be very costly. On the other hand, providers aim to have a balanced work-load with less preserved capacity, yet avoid any monitoring cost or possible punishments. If they do not manage the gap between the actual and ideal resource provisioning, it can negatively affect their reputation and aggregated utility. Meanwhile, the oracle tends to charge higher for the monitoring services without risking a decline in the number of the requests for monitoring that it receives. Yet some important questions remain: how many times and when to ask for quality monitoring, who has to pay for such a service, how much should be paid and how to avoid SLA violations and its possible consequences. To answer the above questions and to optimize the providers' computational resource capacity, quality verification requests and cost of the monitoring, we formulate a dynamic Stackelberg differential game among three agents seeking to maximize their revenue. The Stackelberg differential game is used to study the sequential decision making of cloud provider (leader) for the optimal resource provisioning, cloud requester (follower) for the quality monitoring requests, and oracle (follower) for the monitoring cost. In the designed game, the differential equations capture the dynamic competition and resource provisioning, quality monitoring requests and costs in continues time.

This paper contributes as follows:

1) Developing a novel blockchain-based decentralized model for cloud providers that outsource some parts of their demand which they cannot fulfill on their own. Our proposed model enjoys a multiagent structure, which allows us to introduce a quality verifier agent to ensure the cloud provider's compliance with the SLA. The interaction of an oracle within blockchain for monitoring purposes is innovative.
2) Formulating a three-player dynamic Stackelberg differential game in which players have to make choices about their control variables at various points in time, where PA acts as the leader and RA and VA are the followers. Differential equations are introduced into the game model to characterize the dynamic variations of the end-users' demand. Finally, the optimal solutions are obtained based on the open-loop equilibria of the proposed game.
3) Implementing and evaluating our proposed model using the Solidity language on Ethereum and Web3.js by simulating three real-world cloud providers using our system for 100 days. To the best of our knowledge, there is no research that implements oracles and their practical integration with smart contracts.

Due to the very recently emerging research topic and nonexistence of any similar model, we are not able to compare our model with any other model. In addition to the optimal profit of agents, we also evaluated estimated transactions and costs.

2 Motivational Scenario

For a blockchain-based federation, it is vital to monitor the QoS and ensure that SLA conditions are met, since cloud providers may have an incentive to deviate. This verification is highly desirable considering the multi-tenancy characteristic of cloud services. In this context, to scale the economic benefits and optimize resource utilization, multiple Virtual Machines (VMs) are initiated on the same physical server simultaneously. The performance variation depends on the network load and usage peak from other tenants. Cloud providers try to balance the workloads and achieve the required performance with less preserved capacity, yet they might not be able to supply a consistent performance.

Fig. 1 illustrates a scenario when cooperation among two cloud providers could be problematic. Let us imagine cloud provider A and cloud provider B are using Cloudchain through the following steps:

1) Cloud providers have to supply scalable cloud services with consistent performance for their users with guaranteed SLA. To ensure such scalability and on-promise performance, cloud providers A and B can register themselves in Cloudchain to enjoy the federated services from the available resources.
2) When cloud provider B faces a computing resources deficiency to meet its end users' demand, it can create a request through Cloudchain.
3) Provider A, who has idle servers, accepts the request and leases its computing resources to provider B

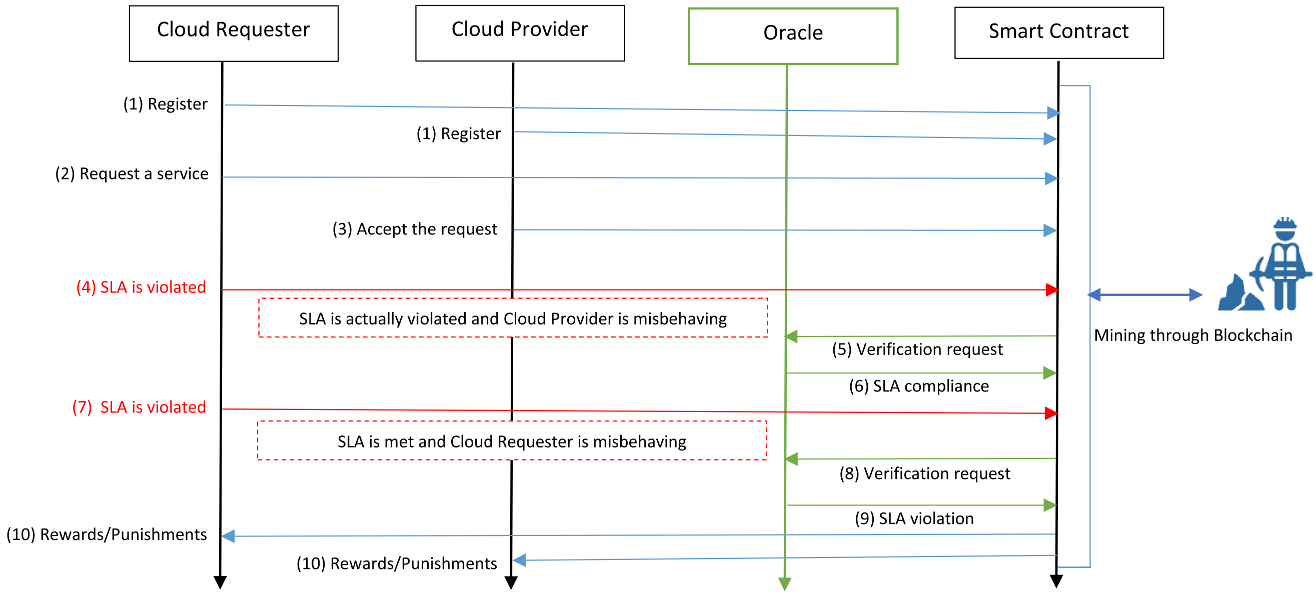


Fig. 1. Cloud providers misbehavior through the federation.

within the smart contract deployed over Cloudchain with a specified SLA, price, terms, and conditions.

4) Two issues might happen that Cloudchain cannot resolve on its own. First, provider A has actually complied with SLA stated in the Cloudchain contract, but provider B claims falsely that provider A violated the SLA conditions and has to be fined.
5) Cloudchain is impotent to oversight and to confirm who is telling the truth due to its inability to communicate with the outside world of the blockchain network. Blockchain can only access information present in a transaction or in the transaction history of the blockchain. Thus, we introduce oracle as a third party to perform the verification and confirm if the SLA is met.
6) The oracle can check the QoS at some cost and report the SLA compliance to Cloudchain.
7) Second issue can arise when provider A has actually compromised the quality but denies the accusation and requests to receive the full deposit from provider B.
8) Cloudchain calls the oracle and initiates a verification request.
9) The oracle confirms that a violation has happened.
10) At the end, considering the terms and conditions of the contract, as well as the verification reports from the oracle, Cloudchain distributes the money and charges the verification cost.

Utilizing the oracle through the steps 4 to 9 is part of our contribution in this paper. In order to fully materialize the oracle as a verifier agent, we first develop a new multi-agent structure and then optimize the cost of using the oracle and trading within Cloudchain.

3 Related Work

This work extends our previous work on Cloudchain [5], a novel model that exploits blockchain to prompt and support interoperability and coopetition among cloud providers over the public Ethereum network. Blockchain is employed to ensure transparency and decentralize the agreements in Cloudchain, but the provided cloud services are supplied out of the blockchain network. So, the blockchain dynamics can neither guarantee nor validate the quality of the supplied service. Therefore, to ensure that the providers comply with the agreements, we need to validate the QoS, even though the agreement is deployed over blockchain. Similar to any other blockchain-based platform, Cloudchain suffers from the most challenging issue, yet to be solved, which is its inability to interact with the outside world. Thus, in this work, we introduce a verifier agent as an autonomous oracle to monitor the quality upon the request of the service requester (RA). We further investigate the revenue maximization strategies among the cloud providers and verifier agent which were not discussed in our previous contribution.

Summaries of the related literature are drawn from three different areas elaborated as follows.

3.1 Game Theory in Cloud Computing

Game theory has been successfully applied in the cloud computing area, for instance for resource allocation and pricing mechanisms, where the interactions of players have to be taken into account [6]. A user-provider interactive approach is taken by Hadji et al. [7], where a Stackelberg game is designed to consider constrained pricing with limited resources offered by a cloud service provider and the optimal user demands. Xu et al. [8] optimized a pricing policy for cloud service providers to better compete with each other under the evolution of the cloud market. Forming a Stackelberg game, the authors applied Q-learning to find out an optimal policy for the leader. Following the leader, the optimal policy for followers will be uncovered. However, the price is the only utility factor considered in these studies and the importance of QoS is somehow neglected. The study by Fan et al. [9] could address the QoS competition issue by considering the market competition among a SaaS (software-as-a-service) provider and a traditional software provider as a differential game. This research analyzes short and long-term competitions for price and dynamic quality between the two firms. The authors found that the cost of software implementation can significantly affect the equilibrium price while quality improvement has a more robust effect.

3.2 Cloud Federation

The literature about cloud providers cooperation focuses on federation formation as coalitional games where capacity and revenue are shared [10]. Coronado et al. had an intensive investigation on federation formation variables among providers, including revenue sharing mechanisms, capacity and cost disparity, and the presence of a big competitor [11]. They defined revenue sharing mechanisms as the most important factor. Among these mechanisms, shapely value and outsourcing models had the least and best performance, respectively. They indicated that collaborating providers can implement a mechanism in which a provider outsources some of its business and gets a percentage of the revenue. The outsourcing model allows the provider to keep some of the revenue of its secured business, even though it is not able to fulfill that business alone. The authors had an insight through the demand peaks and concluded that cloud providers tend to stay in outsourcing collaboration when the demand is high. However, interoperability, trust among providers, and service quality or SLA are not considered in their study. The findings from this work confirm the superiority of outsourcing in terms of maximizing the profit of providers, which is what we are proposing in this paper in addition of having the advantage of coopetition among these providers. The fact that providers tend to collaborate when they face a hike in their demand, reinforces the consideration of a dynamic and long/short-term federation-like blockchain. The challenges of interoperability and trust issues among cloud providers are also addressed by the blockchain platform we propose in this paper.

Wahab et al. [12] focused on the business potential of Web services and addressed the problem of community-based cooperation as a virtual trading market using a Stackelberg game model. Khosrowshahi [13] considered stability and fairness for all web services within a community and offered an applicable mechanism for membership requests and selection of web services. The proposed mechanism used cooperative game-theoretic techniques, particularly Shapley value, core, ϵ-core and convex games. Nonetheless, none of these studies utilized blockchain to form a federation and neither proposed a solution for service quality monitoring for federated services.

Zhao et al. [14] investigated the impact of two factors: energy consumption and SLA violations on degrading the cost-efficiency of data centers and the cloud providers' revenue. The authors developed online VMs placement algorithms as an optimization problem of maximizing revenue from VMs migration and achieved promising results. However, no initiatives are proposed to monitor the SLA violations, specifically when it comes to cooperation and competition among providers. The dynamic and timed decision making strategies are also not considered.

3.3 Blockchain and its Applications

Blockchain has emerged as a distributed database technology building upon a secured list of timestamped transaction records. Its main innovation stems from enabling parties to transact with untrusted parties over a computer network [15]. The blockchain data structure is an ordered list of blocks containing aggregated transactions. Every block is identifiable and linked to the previous block in the chain where the integrity is ensured by cryptographic techniques. Recently, blockchain had a revolutionary impact in corporate governance by offering greater transparency among stakeholders, easier administration, and the creation of an infrastructure for innovative applications where business transactions could be shared in real-time [16]. By leveraging blockchain-enabled smart contracts, we eliminate the need for trust in the federation and reduce barriers of entry, lock-in, and transaction costs, by removing obsolete trust-establishing mechanisms [17]. A smart contract is a piece of code residing on a blockchain and is identifiable by a unique address. Moreover, smart contracts permit creating decentralized applications (DApp) that operate autonomously without any intervention by a system entity.

A few efforts have been made to study the potential of blockchain in real-world applications despite its great potential for businesses to share data and collaborate in a secure and customized manner [15]. According to Tractica, a market research firm, the annual revenue for enterprise applications of blockchain is estimated to reach $19.9 billion by 2025 [18]. The majority of studies about blockchain's application have focused on finance [19], energy [20] and IoT applications [21].

In cloud computing and service industry, to the best of our knowledge, there have been very few related academic initiatives in addition to Cloudchain. Among which, one paid major attention in the energy-aware resource management problem in cloud data centers and developed a robust blockchain-based decentralized resource management framework in order to save the energy consumed by the request scheduler [22]. Moreover, this research further utilizes a reinforcement learning embedded in a smart contract to minimize energy cost. Their simulations based on Google cluster traces and electricity prices showed their method was able to reduce the data centers' cost significantly. Desmaa [17] is a cloud marketplace framework based on blockchain technology. This conceptual framework modeled the interactions between a service provider and a service consumer and tried to overcome problems of conventional marketplace systems, such as barriers of entry and transaction costs. Yet, the outsourcing model with collaboration and competition among cloud providers themselves are not considered in this initiative. Moreover, the providers' profit and the best strategies for utilizing this marketplace are not elaborated nor modeled. Even though the authors developed a prototype, no evaluation and validation against real-world's scenarios were provided.

4 Quality Verification Model Within Cloudchain

4.1 Background: Cloudchain's Smart Contracts

Cloudchain incorporates three types of smart contracts including a set of executable functions and state variables [5]. Similar contracts are proposed in [23] in the context of medical data management. *Contract 1* (*C1*) or Cloudchain Registry (CCR) is a global contract that maps cloud providers identification values (including *Name*, *Reputation Value*, *Computing*

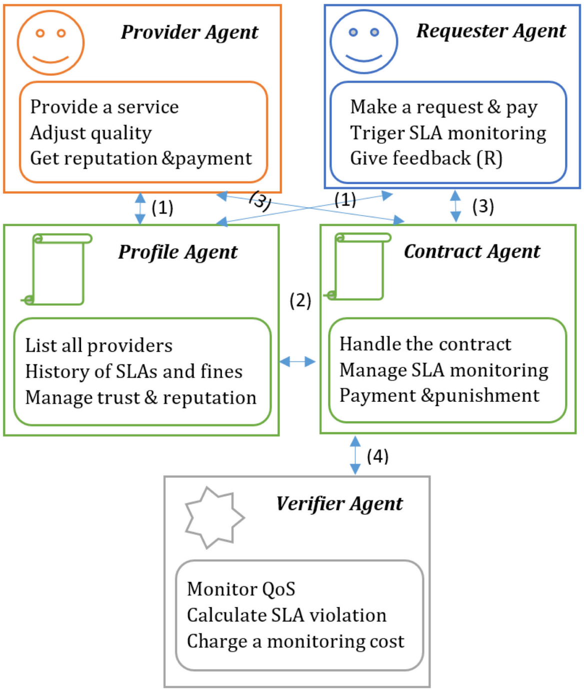


Fig. 2. Multiagent cloud service quality monitoring model.

Capacity and *Storage Capacity*) to their Ethereum address identities (equivalent to public keys). Policies coded into the contract can regulate registering new providers or changing the mapping of the existing ones. The cloud provider registration can be restricted only to certified providers. CCR also maps identities to the Cloudchain Contract (CCC) address on the blockchain, where a special contract regarding each provider profile and list of services is recorded.

Contract 2 (*C2*) denotes Cloudchain Profile (CCP). It holds a list of references to CCC, representing all the participants' previous and current engagements with other nodes in the system. CCP also implements a functionality to enable provider notifications. Ethereum supports an event-based mechanism which permits smart contracts to create an event and signals that a certain action (e.g., an update to profile's data) has been performed. Providers should register their requests in the CCP contract to be propagated and raised to other nodes of providers. Each transaction list stores a status variable. This indicates whether the transaction is newly established, awaiting pending updates and has or has not been completed. This contract is important as it stores the address of all new CCC contracts, without which Cloudchain can simply lose the track of all the contracts.

Contract 3 (*C3*) represents the Cloudchain Contract (CCC). It is issued between two nodes in the system when one node accepts and provides the requested service for the other. The beneficiaries can also complete or cancel the contract. Once the contract is completed or canceled, the contract balance would be transferred to the supplier or requester address respectively, and the contract status would also be updated. There are two approaches to reduce the size of the data as well as the cost of transactions over Cloudchain. The first approach is a common practice for data storage in smart contracts and consists of storing raw data off-chain, and metadata, small critical data, and hashes of the raw data on-chain [24]. However, the selection of off-chain data storage has some concerns regarding the interaction between the blockchain and the off-chain data storage. The other approach is to provide a common glossary among cloud providers to define the generic terms and policies to be referred to in the contract. The members of Cloudchain can join and leave the system anytime by executing specific functions in the smart contracts. Such a flexible membership allows them to supply or demand a service once or multiple times as required.

Notwithstanding the fact that the developed smart contracts can operate as a reliable distributed technology among the cloud providers, its execution environment is self-contained which makes it unable to communicate with the outside world and check if any SLA violation has happened. So, an oracle is required to verify the service quality which is supplied out of the blockchain network. For this purpose, we develop a new multiagent model for cloud service quality monitoring in the following section.

4.2 Multiagent Architecture of the Cloud Service Distributed Model

The proposed model incorporates five agents as follows:

- *The Requester Agent (RA)* and *Provider Agent (PA)* are both cloud providers willing to trade their computing resources;
- *The Registry-Profile Agent (RPA)* and *Contract Agent (CA)* represent CCR-CCP and CCC smart contracts, respectively, with a set of executable functions and state variables; and
- *The Verifier Agent (VA)*, known as oracle, is an agent that verifies real-world occurrences and submits this information to a blockchain to be used by smart contracts.

Fig. 2 provides the interactions among these agents. In step 1, RA and PA should register themselves in RPA where each registered user is assigned with a public key pair. RPA maps identities of RA and PA to the contract agent's (CA) address on the blockchain. It holds a list of references to CA to provide all the participants' previous and current engagements with other nodes in the system with a record of any SLA violation or compliance.

When RA faces a computing resources deficiency, it can submit a request for a service using RPA to create and deploy a CA in the blockchain in step 2. Meanwhile, a rule for providers is set by the requesters to ensure that qualified providers could ultimately receive the task, e.g., reputation value threshold. The first time reputation values can be computed from the customers' ratings given to each provider through online rating platforms (known in recommender systems [25]) and will be updated based on the future ratings given by RA.

CA regulates the interactions between two nodes in the system where one node accepts and provides the requested service to the other in step 3. RA is required to pay a deposit in advance and it is stored in the contract using CA. The beneficiaries can complete or cancel the contract, however, the contract termination and delivery of the requested service have to be confirmed by RA. Once completed or canceled, CA calculates fines to be charged if any exist and the remained contract's balance would be transferred to the RA or PA address respectively, and the contract status would be updated.

Function calls on contracts are transactions, and those which update the contract storage need to be validated by blockchain miners. Once a new block is mined with the newly linked CA, it would be broadcasted to other nodes and the first node that accepts the request should update the respective CA contract.

RA can initiate a quality monitoring request anytime to check if the provider is complying with the SLA conditions during the runtime. The request should be submitted to CA and CA calls VA to perform the verification through step 4. VA checks the prioritized quality attributes of the service using RA credentials and extracts the runtime needed information and pushes it into CA. PA would be penalized if there is any violation of the SLA. A record of the SLA monitoring and penalties will be kept by RPA for future references. RA is required to rate the supplier based on the perceived performance. The process of requesting a service and monitoring its quality among the five agents are elaborated in Algorithms 1 and 2. The agents' decision variables and the details of their trading policies provided in these algorithms will be discussed in the following section.

Time and Space Complexity. The time complexity of Algorithm 1 grows linearly with the number of verification requests (n), therefore it has the time complexity of $O(n)$. Algorithm 2 is independent of the number of requests, and it has the time complexity of $O(1)$. However, running the algorithms over the blockchain affects the complexity. When cloud providers are interacting with Cloudchain, they call various functions. Some of these functions are only reading the states of variables (View functions), and some are changing the states (Transactional functions). View functions, which only read the states and are not called within a transactional function, do not consume gas, and they can only be executed on a local node to get the results. However, in Transactional functions, some states would be changed on Blockchain, therefore new blocks are expected to be created. In this case, all active nodes (m) over Blockchain should run the same code to verify the results of the new block, thus the time complexity would increase to $O(m \times n)$ for Transactional functions in Algorithm 1 and $O(m)$ for Algorithm 2, as the number of nodes increases. The space complexity of Algorithm 1 is $O(n)$ by growing the number of inputs, the required space grows linearly to store the stack of data in a mapping structure. However, when it stores the data on blockchain, there is a triangular pattern ($n(n+1)/2$) stack of the previous versions of the data by every update. In addition, this is also happening on every active node over blockchain ($m \times (n(n+1)/2)$). Thus, the space complexity of Algorithm 1 is $O(m \times n^2)$, and for Algorithm 2 is $O(m)$.

5 Requester, Provider and Verifier Agents Decision Making

Unlike conventional (static) game theoretic models, dynamic models we use in this paper consider the important dimension of time and recognize the competitive decisions that do not necessarily remain fixed. Models involving competition in continuous time are typically treated as differential games, in which critical state variables, e.g., demand, are assumed to change over time according to specified differential equations.

Algorithm 1. Cloud Providers Service Agreements within the Multiagent Model

```
Input: Ether Deposit; Reputation threshold; PA reputation;
    Cloud requester's Etheruem address (RA); Cloud supplier's
    Etheruem address (PA).
procedure ServiceAgreement
    RA makes a service request in RPA
    RPA creates a CA
    RA.SendTo(CA, Ether Deposit)
    CA.Availability = True
    EventLog.Create("New request is available")
    if RA.Reputation $\geq$ Reputation threshold AND
    RA.Accept(CA) then
        CA.Availability = False
        while RA requests a quality verification do        ▷ refer
        to Eq. (8)
            CA calls Algorithm 2
            $N(t)$ += 1
            if Verification.Result = True then
                RA.SendTo(Verifier, $M(t)$)
                PA.PositiveVerification += 1
            else if Verification.Result = False then
                PA.SendTo(Verifier, $M(t)$)
                PA.NegativeVerification += 1
                Assign $F$ to charge PA
                CA.SendTo(RA, $F$)
            end if
        end while
        if CA.Completed then
            EventLog.Create("CA is completed")
            ContractDeposit = CA.TotalAmount
            CA.SendTo(PA, ContractDeposit)
            EventLog.Create("Fund is transferred to the Cloud
            supplier")
            RA.UpdateReputation(PA)
        end if
    end if
end procedure
```

Algorithm 2. Verification Process

```
Input: CA Ethereum address; $M(t)$; $\epsilon$; VA Etheruem address
(oracle).
Output: Verification.Result                              ▷ Boolean
procedure VerificationProcess
    VA retrieves CA terms and monitor the cloud service
    VA verifies the quality of the provided service based on
    CA.SLA
    if $(\bar{\phi} - \phi(t)) <= \epsilon$ then
        Verification.Result = True
    else
        Verification.Result = False
    end if
    Calculate $M(t)$ to charge CA                ▷ refer to Eq. (15)
end procedure
```

Our RA, PA, and VA agents aim to maximize their profit within our proposed multiagent and blockchain-based federation. RA is facing a peak time and is going to request some VMs from other federation's members, and PA has some idle servers and is willing to rent them out with the price offered by RA. For simplicity and without losing generality, we will focus on a single VM type, with $\bar{\phi}$ denoting the desired capacity and the process rate of VM instances that can be hosted by PA which guarantees to meet the SLA even during the peak time. In fact $\bar{\phi}$ is what RA is paying for, while ϕ is the actual preserved capacity that PA assigns for RA considering the fact that its other customers might use less and it can assign the extra capacity to RA when needed. Therefore, PA controls its optimal capacity assignment ϕ. Since RA might experience a QoS degradation or sense a violation of SLA, it has the right to initiate a monitoring verification request. However, this request can be costly as it has to pay if there is no SLA violation. Thus, RA is required to decide on the number of verification requests to make, denoted by the control path $N(t)$ (a control path is a vector of control or decision variables). On the other hand, VA has to decide on its control path representing the optimal pricing $M(t)$ which alters its verification demands and final revenue.

We formulate the profit maximization, service trading, and quality assurance problems as a Stackelberg differential game as follows.

- Players: There are three players: PA acts as leader; and VA and RA act as followers. We assume that the decision making of the followers are simultaneous and they use each others' control variable as input of their models.
- Strategy space: PA can choose the optimal capacity control path $\phi(t)$ to maximize its payoff by observing the cost and numbers of the requests for monitoring in response to the capacity. VA sets an optimal price control path $M(t)$ to charge for the quality verification by considering the given capacity which also affects the number of the requests from RA. RA controls its verification requests $N(t)$ to ensure the service quality by considering the given $\phi(t)$.
- State: The end users' demands and the quality monitoring demands are the system states of PA, RA, and VA, respectively.

In the Stackelberg differential game, both the leader and followers try to maximize their own payoffs, which are the integration of instantaneous payoffs over the time horizon $[0, T]$, by controlling their control paths which are their decision variables to be committed to for the whole time horizon. Similar to a statistic Stackelberg, to obtain the equilibrium of a Stackelberg differential game, we use backward induction and solve the problem for the follower first. A list of the notations is provided in Table 1.

TABLE 1
Notations Used in Stackelberg Differential Game

j/i	Index of a provider/quality attribute from the set k/I
M	Quality monitoring cost
N	Number of monitoring verification requests
$r/p/v$	Requester/provider/verifier agents
G/C	Cost of gas/capacity
F	Fine payment of PA to RA due to SLA violation
W	The amount of the cumulated gas for each block
$W^{'}$	The amount of required gas for each transaction
X	Number of the transactions over the blockchain
Y	Maximum number of the initiated verification requests
ω	Rate of transactions arrival for an agent in $[0-T]$
$\omega^{'}$	Weight of the monitoring request arrival from RA
Γ	Rate of the block generation for miners
$\phi/\phi^{'}/\bar{\phi}$	Provider's active/mining/maximum capacity
P	Price per VM
R	Reputation value of RA and PA in the range of $[0-1]$
Rw	Reward value of mining
τ	Block propagation time
η	Rate of the impact of W over τ
ψ	Rate of end users' demands increase due to higher quality for RA
θ	Rate of end users' demands increase due to higher R_p
β	Quality sensitivity of the end users
γ	Price sensitivity of RA and PA for monitoring service
δ	Demand decay rate
μ	The amount of VMs
Q	Quality attributes of a cloud service in the set I
$\lambda/\Lambda/\Theta$	Adjoint variables

5.1 Preliminaries

This section explains some of the required elements to formulate our game. We first make an assumption that is a rule for differential games.

Assumption. *Each player has perfect knowledge of:*

- *The dynamic state function determining the evolution of the demand, and the control paths of the three players.*
- *The payoff functions.*
- *The initial demand states at time zero.*

The analysis of differential games relies profoundly on the concepts and techniques of optimal control theory [26]. Definition 4.1 provides some relevant points on this regards.

Definition 1. *The open-loop strategy spaces of RA, VA and PA are respectively defined as:*
$\tilde{N} = \{N(t)|N(t) \text{ measurable on } [0,T], N(t) \geq 0 \text{ for all } t \in [0,T]\}$
$\tilde{M} = \{M(t)|M(t) \text{ measurable on } [0,T], M(t) > 0 \text{ for all } t \in [0,T]\}$
$\tilde{\phi} = \{\phi(t)|\phi(t) \text{ measurable on } [0,T], 0 > \phi(t) \geq \bar{\phi} \text{ for all } t \in [0,T]\}$. *The strategy profile $(N^*(t), M^*(t), \phi^*(t))$ is an open-loop Stackelberg equilibrium if, for RA, VA and PA, each of them is optimal control strategies given others' strategies.*

It is assumed that cloud providers can participate in the mining to earn some rewards if they have spare computing resources. In the mining race, miners have to compete to solve proof of work and propagate the block to reach consensus. The new blocks' generation follows a Poisson process with a constant rate $\frac{1}{\Gamma}$ throughout the whole blockchain network [27]. Before the race, miners collect their selected pending transactions into their blocks with a total gas amount of $\sum_{j=1}^{k} W_j$. Gas is a proportional amount that Ethereum pays to motivate the miners to participate in the mining process. When miner j propagates its block to the blockchain for consensus, the time for verifying each transaction is affected by the total size of the transactions W_j. The first miner j who successfully has its block achieves consensus will be rewarded based on the amount of the assigned capacity $\phi^{'}_j$. The amount of the reward can be computed by the following proposition.

Theorem 1. *The miner j's expected reward $Rw_j(\phi^{'}_j)$ is:*

$$Rw_j(\phi^{'}_j) = Rw_j\phi^{'}_j e^{-\frac{1}{\Gamma}W_j\eta_j}. \tag{1}$$

Proof. The expected reward of mining is: $Rw_j\mathbb{P}_j(\phi_j^{'}, W_j)$ where $\mathbb{P}_j(\phi_j^{'}, W_j)$ is the probability that miner j receives the reward by contributing a block. To win the reward, the miner must perform a successful mining and instant propagation. However, the miner may fail to obtain the reward if its new block does not achieve consensus as the first. This kind of mined block that cannot be added to the blockchain is called orphaned block. The block containing a larger size of transactions has a higher chance of becoming orphaned since a larger block requires more propagation time, thus, causing a higher delay for consensus. As the arrival of new blocks follows a Poisson distribution, miner j's orphaning probability, $\mathbb{P}_j^0$, can be approximated as:

$$\mathbb{P}_j^0 = 1 - exp\left(-\frac{1}{\Gamma}\right)\tau_j. \tag{2}$$

It is safe to assume that miner j's block propagation time τ_j is linear with the size of transactions in its block, $\tau_j = W_j\eta_j$, where η_j is a constant that reflects the impact of W_j over τ_j. Therefore, we obtain the reward probability from Eq. (2) as follows:

$$\mathbb{P}_j(\phi_j^{'}, W_j) = 1 - \mathbb{P}_j^0 = \phi_j^{'} e^{-\frac{1}{\Gamma}W_j\eta_j}. \tag{3}$$

By multiplying Eq. (3) by the reward value, we obtain Eq. (1). □

Definition 2. *To model the transactions' distribution, we use the compound Poisson process, which is a generalization of the Poisson process where each arrival is weighted according to a distribution. The compound Poisson process represents better the transactions dynamics. The assumption is that transactions sent to the blockchain follow a Poisson process, but the amount of gas they require follows a compound Poisson process. The reason is that the difference between the amount of gas is based on the complexity of the transaction, for example, the creation of a contract requires a much higher amount of gas than updating the contract. Therefore, the probability of the required gas by X_j transactions occurring in $[0 - T]$ follows an exponential distribution as follows:*

$$\mathbb{P}_j(X) = \frac{e^{-\omega T}(\omega T)^{X_j}}{X_j!}. \tag{4}$$

We further require another distribution function to formulate the cost and penalty/reward of the quality monitoring services provided by VA. To do so, we require to consider how the history records of transactions and the reputation values of the providers can influence the number of the requests for quality verification. This formulation is provided in the below definition.

Definition 3. *Let us define Y as the maximum number of the verification requests that an agent j can initiate. If the reputation value of an agent who is providing a service is high, it is likely that there will be less number of the verification requests. However, if the reputation value of RA is high it is more likey to have more numbers of verifications to ensure that quality will remain reliable and the reputation stays untouched. Thus, the most requests will be initiated if the reputation value of RA is high and PA is low. We propose a similar distribution function given by Eq. (4), since the initiation of monitoring requests and the amount of cost $M(t)$ or the received reward (penalty F paid by PA) follows the same compound Poisson process which is tight with the agents' reputation values:*

$$\mathbb{P}_j(Y) = \frac{e^{-\omega' T}(\omega' T)^{Y_j(1-R_p)R_r}}{(Y_j(1-R_p)R_r)!}. \tag{5}$$

5.2 Problem Formulation and Open-Loop Equilibrium of RA (Follower)

We first discuss the problem of the optimal number of verification requests $N(t)$ for RA, to get the open loop equilibrium solution. Adopted from [28] and [29], we define the end users' demand using the Cobb-Douglas function that captures the demand elasticities and variations specific for each user of a cloud service, $D = \mu\ P^{-\alpha}\ Q^{\beta}$. The two variables α and β are the users' sensitivities towards the price and quality, respectively.

As presented in Algorithms 1 and 2, we assume that if $(\bar{\phi} - \phi(t)) <= \epsilon$, then PA is complying with the SLA and there will be no penalty of F and no verification cost $M(t)$ for PA. Here, ϵ is a very small number that PA is allowed to disobey due to the very dynamic context of cloud service attributes. But, if $(\bar{\phi} - \phi(t)) > \epsilon$, then RA will not pay for $M(t)$ and will be rewarded the penalty value of F. The provider's cost is usually considered to be quadratic in the literature [30], a convex cost term can prevent aggressive behavior of a certain provider which can result in a monopoly market.

The requester agent acts as a follower of PA by relying on the provided capacity and simultaneously receiving the monitoring cost from VA to adjust its optimal number of the verification monitoring requests $N(t)$, while considering the evolve of its end users' demand state. Thus, it tries to maximize its profit according to the following function:

$$\begin{aligned} \text{Maximize} \quad & RA(N(t), D_r(t), M(t), \phi(t), t) \\ & = \int_0^T e^{\rho t}\{P_r D_r(t) - \bar{\phi}P_p^2 + Rw_r(\phi_r^{'}) - \mathbb{P}_r(X)W_r^{'}G_r \\ & - \mathbb{P}_r(Y)(M(t) - F)(\bar{\phi} - \phi(t))N(t)\}dt \\ \text{subject to} \quad & \dot{D}_r(t) = (N(t)\phi(t))^{\beta}\psi - \delta_r D_r(t), \end{aligned} \tag{6}$$

$Rw_r(\phi_r^{'})$, $\mathbb{P}_r(X)$ and $\mathbb{P}_r(Y)$ are borrowed from Proposition 4, Definition 4.2 and Definition 4.3. ρ denotes discounted factor in our discounted-utility model, in which it is assumed that the instantaneous utility each period depends solely on profit in that period, and that the utilities from streams of profit are discounted exponentially.

The demand state dynamics is defined with $\dot{D}_r(t)$ which increases with the number of capacity monitoring and ensuring the service quality with the power of β as the end user sensitivities towards the quality at the rate ψ. It also decays at the rate δ_r in which users switch to another provider.

Definition 4. *For RA, the number of monitoring requests' strategy $N(t)^*$ is optimal if the following inequality holds for all feasible control $N^*(t) \neq N(t)$, $RA(N(t)^*, D_r(t)) \geq RA(N(t), D_r(t))$.*

In order to get the equilibrium solution of the optimization problem in Eq. (6), we need to construct the Hamiltonian system of the RA's problem. Equilibrium strategies in the open-loop structures can be found by solving a two-point boundary value problem for ordinary differential equations derived from the Pontryagin maximum principle in Hamiltonian functions. Here, the equilibrium solution for RA is the solution of the differential game, and also is the Stackelberg equilibrium solution for RA as a follower. The Hamiltonian system of RA is as follows:

$$\begin{aligned} H_r(t) =& P_rD_r(t) - \bar{\phi}P_p^2 + Rw_r(\phi_r^{'}) - \mathbb{P}_r(X)W_r^{'}G_r \\ &- \mathbb{P}_r(Y)(M(t)r - F)(\bar{\phi} - \phi(t))N(t) \\ &+ \lambda(t)((N(t)\phi(t))^{\beta}\psi - \delta_rD_r(t)). \end{aligned} \tag{7}$$

The adjoint variable or shadow price (λ) associated with a particular constraint is the change in the optimal value of the objective function per unit increase in the right-hand-side value of that constraint, all other problem data remaining unchanged. The economic interpretation of $\lambda(t)$ is the value of an additional unit of the end users' demand for RA. For a given $N(t)$, $\lambda(t) > 0$ implies that RA benefits from the current demands. With a zero shadow price $\lambda(t) = 0$, RA does not take into account the impact of the price on future users' demands. On the other hand, when $\lambda(t) < 0$, RA has no motive to sacrifice current profits for future profits by paying the cost of quality monitoring, so that it will no longer increase $N(t)$. The final solution is obtained in the following Theorem.

Theorem 2. *Knowing the fact that verification cost is paid by RA unless there is a violation of the SLA by PA, in which PA incurs $M(t)$ and F, the optimal number of monitoring requests is given by:*

$$N(t)^* = \begin{cases} \left(\frac{-\mathbb{P}_r(Y)M(t)(\bar{\phi}-\phi(t))}{\lambda(t)\beta\psi\phi(t)^{\beta-1}}\right)^{\frac{1}{\beta-1}} & \text{if } (\bar{\phi} - \phi(t)) \ <= \epsilon \\ \left(\frac{\mathbb{P}_r(Y)F(\bar{\phi}-\phi(t))}{\lambda(t)\beta\psi\phi(t)^{\beta-1}}\right)^{\frac{1}{\beta-1}} & \text{if } (\bar{\phi} - \phi(t)) \ > \ \epsilon \end{cases}, \tag{8}$$

and $\lambda(t)$ is given by:

$$\lambda(t) = \frac{P_r}{\rho - \delta_r}(1 - exp((\rho - \delta_r)(t - T)). \tag{9}$$

Proof. As proven in the optimal control theory, the optimal control strategy of the original problem must also maximize the corresponding Hamiltonian function. According to the Pontryagin's Maximum Principle (PMP), a control constitutes an open loop equilibrium to the problem in Eq. (7), and $D_r(t)$ is the corresponding state trajectory, if there exists a costate function $\lambda(t)$ such that the following relations are satisfied,

$$\begin{aligned} \frac{\partial H_{RA}(t)}{\partial N(t)} =& \mathbb{P}_r(Y)(M(t)r - F)(\bar{\phi} - \phi(t)) \\ &+ \lambda(t)\beta\psi(N(t)\phi(t))^{\beta-1} = 0 \end{aligned} \tag{10}$$

$$\dot{\lambda}(t) = \rho\lambda(t) - \frac{\partial H_{RA}^*(t)}{\partial D_r(t)} = \lambda(t)(\rho - \delta_r) - P_r, \lambda(T) = 0, \tag{11}$$

where Eq. (11) is the adjoint equation to describe the dynamics of a costate variable. In the case that the strategy space $\tilde{N}$ does not depend on the system state D_r, the maximized Hamiltonian function H_{RA}^* on the right hand side of Eq. (11) can be replaced by the original H_{RA}. When only one boundary condition is specified as $D_{r0}(t) = D_r(0)$, the free-end condition is used as $\lambda = 0$ at $t = T$. Solving the differential equation of Eq. (11) can lead us to the corresponding costate function. By solving Eq. (10), we can obtain the optimal $N(t)^*$ given in Eq. (8). □

5.3 Problem Formulation and Open-Loop Equilibrium of VA (Follower)

To formulate the optimal pricing control $M(t)$ problem for VA and to get the open loop equilibrium solution, we require to define the dynamic variation of the state of verification demand. A major part of dynamic pricing research originates from the Bass new service diffusion model, which was later enriched by incorporating price sensitivity to allow a dynamic pricing examination [31], [32]. We modify this model to elaborate on the new concept of verification demand in our model as described in the following definition.

Definition 5. *Let $V(t)$ denotes the total number of the verification requests initiated by RA at time t for each quality attribute, given an I-tuple of QoS attributes $Q_1, Q_2, \ldots, Q_I$ with an index of i. The verification state evolves based on the external factor of capacity discrepancy and the internal factor of price as follow:*

$$\dot{V}(t) = dV(t)/dt = (M(t)V(t) + (\bar{\phi} - \phi(t)))(1 - \gamma M(t)), \tag{12}$$

where the positive parameter γ measures the providers' sensitivity to the verification price.

To obtain a suitable dynamic pricing strategy, VA observes the number of the verification requests from RA and provides a response to the announced capacity control of PA. Therefore, VA tries to maximize its profit by the following Eq. (13) which is subject to Eq. (12):

$$\begin{aligned} &\text{Maximize} \quad VA(M(t), N(t), V(t), t) \\ &\qquad = \int_0^T e^{\rho t}\{(M(t) - C_v^2)V(t) - \mathbb{P}_v(X)W_v^{'}G_v\}dt \\ &\text{subject to:} \quad \dot{V}(t) = (M(t)V(t) + (\bar{\phi} - \phi(t)))(1 - \gamma M(t)). \end{aligned} \tag{13}$$

The Hamiltonian system is given as below.

$$\begin{aligned} H_{VA}(t) =& (M(t) - C_v^2)V(t) - \mathbb{P}_v(X)W_v^{'}G_v + \\ & \Lambda(t)(M(t)V(t) + (\bar{\phi} - \phi(t)))(1 - \gamma M(t)). \end{aligned} \tag{14}$$

Theorem 3. *The optimal monitoring cost is given by,*

$$M(t)^* = \frac{1}{\gamma} - \frac{(\bar{\phi} - \phi(t))}{V(t) + \Lambda(t) + 1}, \tag{15}$$

where $\Lambda(t)$ is given by:

$$\Lambda(t) = \left(\frac{\gamma(\bar{\phi} - \phi(t)) + V(t) - C_v^2V(t)\gamma}{V(t)(\rho\gamma + 1)}\right)\left(1 - exp(\rho + \frac{1}{\gamma}\right)(t - T)). \tag{16}$$

Proof. Similarly, the necessary optimality conditions for VA can be derived according to PMP as follows:

$$\frac{\partial H_{VA}(t)}{\partial M(t)} = 0$$
$$V(t) + \Lambda(t)(V(t)(1 - \gamma M(t)) - \gamma(M(t)V(t) + (\bar{\phi} - \phi(t)))) = 0. \tag{17}$$

By solving Eq. (17), we can obtain the optimal price given by Eq. (15). When the optimal control depends on the system state, it has to be replaced in the original Hamiltonian system in Eq. (14) to achieve $H^*_{VA}(t)$ and to be used for calculation of the adjoint variable $\Lambda(t)$.

$$\begin{aligned} \dot{\Lambda}(t) &= \rho\Lambda(t) - \frac{\partial H^*_{VA}(t)}{\partial V(t)}, \Lambda(T) = 0 \\ \dot{\Lambda}(t) &= \Lambda(t)\left(\rho + \frac{1}{\gamma}\right) - \frac{\bar{\phi} - \phi(t)}{V(t)} - \frac{1}{\gamma} + C_v^2. \end{aligned} \tag{18}$$

□

5.4 Problem Formulation and Open-Loop Equilibrium of PA (Leader)

For each capacity path $\phi(t) \in \tilde{\phi}$ PA announces, there is a corresponding $N^*(t) \in \tilde{N}$ and a $M^*(t) \in \tilde{M}$. PA takes the VA and RA's best responses into consideration when solving the optimization problem. The PA's optimization problem is given by:

$$\begin{aligned} \text{Maximize } & PA(\phi_s(t), R(t), D_s(t), \lambda(t), \Lambda(t), \Gamma(t), M^*(t), N^*(t), t) \\ &= \int_0^T e^{\rho t}\{P_p D_p(t) - \phi(t)C_p^2 + Rw_{jp}(\phi_p^{'}) \\ &- \mathbb{P}_p(X)W_p^{'}G_p - \mathbb{P}_p(Y)(M(t)^* + F)(\bar{\phi} - \phi(t))N(t)^*\}dt \end{aligned} \tag{19}$$

$$\text{subject to} \quad \begin{cases} \dot{D}_p(t) = (R_p - F(\bar{\phi} - \phi(t)))^{\beta}\theta - \delta_p D_p(t) \\ \dot{\lambda}(t) = \lambda(t)(\rho - \delta_r) - P_r \\ \dot{\Lambda}(t) = \Lambda(t)(\rho + \frac{1}{\gamma}) - \frac{\bar{\phi} - \phi(t)}{V(t)} - \frac{1}{\gamma} + C_v^2. \end{cases} \tag{20}$$

Compared to RA and VA, the Hamiltonian function of PA in the Stackelberg differential game is more complex since the maximization of the payoff of PA also needs to consider the dynamics of costate variables of RA and VA as the additional state constraints besides the system state constraints. In this case, similar to the introduction of a costate variable for the system states in the follower's Hamiltonian function, the costate variables for both the system states and costates of the followers are needed in the Hamiltonian function of PA as leader.

$$\begin{aligned} H_{PA}(t) =& P_p D_p(t) - \phi(t)C_p^2 + Rw_{jp}(\phi_p^{'}) - \mathbb{P}_p(X)W_p^{'}G_p \\ &- \mathbb{P}_p(Y)(M(t)^* + F)(\bar{\phi} - \phi(t))N(t)^* \\ &+ \varTheta_1(t)(\theta(R_p - F(\bar{\phi} - \phi(t)))^{\beta}) - \delta_p D_p(t)) \\ &+ \varTheta_2(t)(\lambda(t)(\rho - \delta_r) - P_r) \\ &+ \varTheta_3(t)(\Lambda(t)\left(\rho + \frac{1}{\gamma}\right) - \frac{\bar{\phi} - \phi(t)}{V(t)} - \frac{1}{\gamma} + C_v^2\right). \end{aligned} \tag{21}$$

TABLE 2
Provider's Estimated Transactions and Costs Based on the Proposed Scenarios

Cloud providers	Amazon	C-Link	Alibaba
Reputation	0.9	0.6	0.8
Price	0.0058	0.025	0.0125
Requests	n/a	14	17
Consumed gas*	1,739,596	32,022,933	36,254,668
Gas Price	15	12	11
Gas cost (G)†	26,093,940	384,275,203	398,801,348
Gas Cost (USD)‡	\$12.06	\$256.50	\$248.45
Transaction Delay(s)§	27-66	27-4000	27-5459

*$\sum_{Y=1}^{y} W^{'}$
†*Total Gas×Gas Price*
‡*Average*
§*Time range of each transaction in seconds*

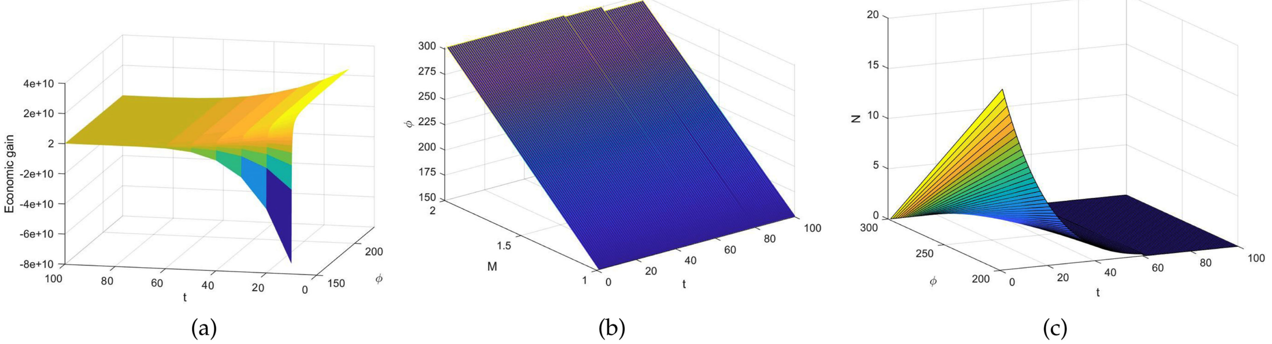
Similarly, the necessary optimality conditions for PA can be derived through the PMP. Due to the concavity of Hamiltonian function with respect to $\phi(t)$, we can obtain $\phi^*(t)$ for the leader which could be$\varTheta$ obtained from $\frac{\partial H_{PA}(t)}{\partial \phi(t)} = 0$, and denoted as $\phi^*(t) = g_p(M^*(t), N^*(t), \lambda(t), \Lambda(t), \varTheta_1(t), \varTheta_2(t), \varTheta_3(t), D_p(t), V(t), t)$. We further have the following conditions:

$$\begin{aligned} \dot{\varTheta}_1(t) &= \rho\varTheta_1(t) - \frac{\partial H_{PA}(t)}{\partial D_p(t)} = \varTheta_1(t)(\rho - \delta_p) - P_p \\ \dot{\varTheta}_2(t) &= \rho\varTheta_2(t) - \frac{\partial H_{PA}(t)}{\partial \lambda(t)} = -\varTheta_2(t)\delta_r \\ &- \left(\tfrac{1}{1-\beta}\right)\lambda^{\frac{\beta}{1-\beta}}\left(\tfrac{\mathbb{P}_j(Y)F(\bar{\phi} - \phi^*(t))}{\beta\psi\phi^*(t)^{\beta-1}}\right)^{\frac{1}{\beta-1}}\mathbb{P}_p(Y)(M^*(t) + F)(\bar{\phi} - \phi(t)) \\ \dot{\varTheta}_3(t) &= \rho\varTheta_3(t) - \frac{\partial H_{PA}(t)}{\partial \Lambda(t)} = \\ &\rho(\varTheta_3(t) - 1) + \frac{1}{\gamma} + \frac{\mathbb{P}_p(Y)(\bar{\phi} - \phi^*(t))^2 N(t)^*}{V(t) + \Lambda(t) + 1}, \end{aligned} \tag{22}$$

with boundary conditions $\varTheta_1(T) = 0$ and $\varTheta_2(0) = \varTheta_3(0) = 0$. We impose $\varTheta_1(T) = 0$, because $D_r(T)$ is free to move and impose $\varTheta_2(0)$ and $\varTheta_3(0)$ to be 0, because our problem is controllable and initial state depends on $\phi(0)$. Replacing $\phi^*(t)$ into Eq. (16) along with differential equations in Eq. (22) constitutes a system of five differential equations which, along with the boundary conditions, imply a solution; however is difficult to obtain analytical solutions for that (you can refer to [33] for a discussion of the complexity of the solutions to a similar system). Yet, we analyze all the variables and the system behavior in Section 5.

Theorem 4. *For the formulated Stackelberg differential game, the candidate strategy profile $(N^*(t), M^*(t), \phi^*(t))$ is indeed an open-loop Stackelberg equilibrium.*

Proof. It is straightforward that the construction of strategy profile $(N^*(t), M^*(t), \phi^*(t))$ satisfies all the necessary conditions as it followed the PMP conditions. The following arguments constitute the sufficient conditions for optimality. Since the Hamiltonian function H_{RA} is strictly concave and continuously differentiable with respect to $N(t)$ for all $t \in [0, T]$, the necessary optimality condition in Eq. (10) uniquely determines a candidate optimal control path $N(t)^*$ as a function of the observed verification pricing strategy $M(t)$, capacity strategy $\phi(t)$ and the system state



(a) (b) (c)

Fig. 3. (a) Optimal capacity of Amazon (PA) emerging towards equilibrium, (b) Dynamic pricing strategy of VA in the equilibrium, (c) Optimal quality verification requests for Century Link (RA) in the equilibrium.

$\dot{D}_r(t)$, and the costate $\lambda(t)$. In a similar way, due to the strict concavity of Hamiltonian function H_{VA} with respect to $M(t)$, and H_{PA} with respect to $\phi(t)$, PMP provides not only necessary conditions but also sufficient conditions for optimality of $M^*(t)$ for VA and $\phi^*(t)$ for PA. According to the stated conditions, we can conclude that the obtained strategy profile is indeed an open-loop Nash equilibrium. □

6 Implementation and Evaluation

We implemented our proposed blockchain-based quality monitoring prototype on Ethereum using Solidity (version 0.4.25), the script language on Ethereum and Web3.js. This program is available open-source in Github.[1] The program was written with the main concern of the minimum consumption of gas per each transaction and was tested using remix,[2] an online IDE for Solidity. The gas price unit is in gwei, which is 1×10^{-9} ether. In our implemented prototype, we used solidity structures and variables to store the provider's data and requests inside the contracts. Meanwhile, each transaction is logged with a summary using an event to make it easily accessible for the other providers (blockchain nodes) to track new transactions. Once a new transaction with a specific event (e.g., new request) is created, other providers can call the contract to get more information and/or change contract stored data. To make the simulation more realistic, we followed up all the contract transactions from registering up to confirmation of the contract completion and assigning a reputation.

For the sake of representation, we assumed three real-world cloud providers (Amazon (PA), Alibaba cloud (RA), and Century Link (RA)) using the system for a duration of 100 days to investigate their economic gain through the Stackelberg differential game. The scalability of our system for a higher number of providers is not questioned since the Ethereum platform is proven to be scalable.[3] We simulated Alibaba and Century Link as cloud requesters who make 17 and 14 requests of service, respectively. The on-demand services' prices are borrowed from the providers' websites and their ratings are collected through the Gartner's dataset.[4] The collected real-world data, simulated number of requests and simulated results of total gas consumption, gas price, gas cost (at the time of writing this paper), and transactions delays are shown in Table 2.

The gas price that providers choose to pay for each transaction can affect the speed of processing their transactions to be approved since miners choose the most profitable transactions to include in their block. We adopt the optimal gas price formulated in our previous study [5]. As Table 2 depicts, the obtained gas consumptions of cloud service requesters are much higher than those that answer these requests and supply these services. This is why Alibaba has the most and Microsoft the least gas consumption. To compute the transaction delays, we had two options: 1- connecting as a node to Ethereum network and collect data to estimate the delays based on the given gas price, 2- using provided online tools that present live monitoring and sound predictions. To save time and get sufficiently large data to obtain an accurate delay range, we followed the second option and used ETH gas station[5] which estimates price and time based on the last 1,500 blocks created each time. Thus, for each transaction, we tested different prices in different time slots to obtain an approximate range of delay depending on the traffic of the Ethereum network. Since there is no time-dependent profit maximization model similar to our proposal, not even in traditional centralized federations or related experiments to be compared to, only the results of our model are reported.

Fig. 3a illustrates the optimal $\phi(t)$ for Amazon, where $\bar{\phi} = 304$ according to the speed attribute mentioned in the SLA terms of Amazon. It can be easily noticed that during the first half period of T, it is crucial to preserve a capacity close to the desired capacity all the time, otherwise, Amazon incurs a huge loss. The surprising point is that Amazon can cheat over the preserved capacity after $t = 60$, since it does not influence its profit. In this situation, Amazon will not reserve the whole resource for Century Link, and if the request consumes extra computing resource than the reserved one, the workload will be shared with other tenants. In this way, there is a minimal risk of penalty and monitoring surcharge as the number of Amazon's customers and tenants grow over time. It should be noted that the pattern was similar for both followers.

The optimal pricing of VA in response to $\phi(t)$ is provided in Fig. 3b. According to our findings, the impact of timing in dynamic pricing is very minimal, meanwhile capacity is strongly correlated with monitoring price. This strong correlation was expected according to Eq. (15) in Theorem 3.

[1] https://github.com/kavehbc/Cloudchain
[2] https://remix.ethereum.org
[3] Scalability is in terms of the number of users, however, it is worth mentioning that the current version of Ethereum suffers handling a huge number of transactions efficiently.
[4] https://www.gartner.com/reviews/market/public-cloud-iaas
[5] https://ethgasstation.info

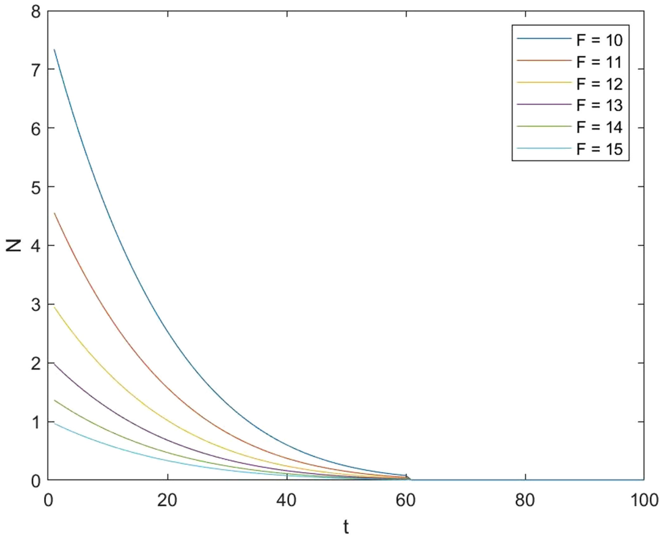


Fig. 4. Optimal quality verification requests for Alibaba (RA) in the equilibrium.

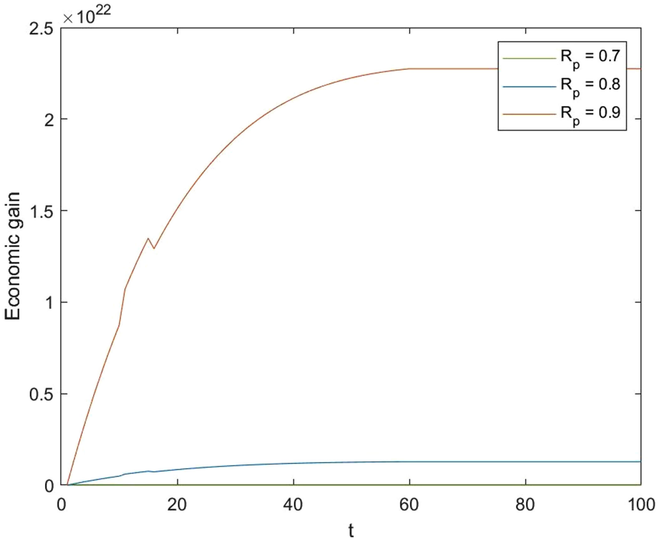


Fig. 5. The impact of the Amazon's reputation over Alibaba's profit.

As the capacity increases, the computational cost and time for VA also climb. Consequently, VA has to enhance its price to be profitable. Another reason could be the low number of the monitoring requests, initiated when the capacity is almost desired, as shown in Fig. 3c. This is also aligned with Eq. (8) in Theorem 2 where the number of verification requests has to be much lower if the difference of the provided capacity is less that ϵ. Century Link had the most number of requests during the first half of the period with very high intensity at the beginning where $\phi(t)$ was low. For the second half period, Century Link is well informed about the quality and the results of the verification from VA, so the number of the requests is almost flattened. This behavior of Century Link now justifies why Amazon can cheat over the quality after $t = 60$. Alibaba Cloud also showed a similar pattern, though the scale was different. Alibaba had a higher number of requests for verification due to its higher reputation value and number of transactions. It is worthy to mention that this response is given to a finite time, it could be different if we assume they will be collaborating for an infinite time.

We further investigated the effect of the penalty value and the reputation value of PA over RA's profit and optimal control. As shown in Fig. 4, the number of the requests for verification starts declining unexpectedly, as penalty F for Amazon (meaning reward for Alibaba) is getting higher. However, based on Eq. (8), we expected the number of verification requests to be increased in case of undesired capacity allocation. This means that with a higher penalty, Amazon will not risk over $\phi(t)$, so the probability of earning a reward is low. Whereas, it is pretty much probable that Alibaba ends up with the monitoring cost to be paid. So, imposing a higher penalty to the provider agent, will increase the capacity and decrease the quality verification requests' equilibria. The reputation value of PA has a significant effect on the profit of RA. As shown in Fig. 5, Alibaba with a reputation of 0.8 gain most if the reputation of PA is higher than Alibaba itself. If the reputation value of Amazon drops to less than 0.8, it is not economically justified to outsource Alibaba's demand to it. This highlights the effect of the users' satisfaction over the demand evolution and economic gain over time. The reputation threshold is certainly less for Century Link with lower reputation value.

7 Conclusion

To overcome the issue of traditional federations of cloud providers and compromised QoS, this paper proposed a multiagent blockchain-based quality monitoring model. In our proposed model, a multiagent approach was taken where an oracle plays the role of a verifier agent to evaluate the service quality whenever is called through the smart contract agents deployed on the blockchain. A Stackelberg differential game was designed to formulate the best strategies of resource provisioning, the number of quality verification requests and the monitoring price for the provider, requester and verifier agents, respectively. The system was implemented using Solidity on Ethereum and was simulated for resource trading among three real-world cloud providers. It was found that at the beginning, the provider has to preserve the desired amount of capacity to satisfy the required quality even throughout the peak-times. However, it is not economically justified to make this reservation for the last periods of its contracts' time. Such a resource provisioning impacted the verifier pricing strategy and the number of requests. RA asks for more verification during the first half period of the contract when the preserved capacity is low. The reputation of PA elevated the profit of RA, whereas, it negatively affected the reputation of PA when it is lower than RA. Furthermore, a higher penalty raised the capacity and reduced the number of verification requests at the equilibrium. The developed system was proven to be economical for cloud beneficiaries and valuable in transparency and preventing the SLA violation.

Our proposed model in this research assumed having a single trusted oracle to perform the verification. However, having a fully-blinded trust on a single third-party may hinder the reliability and efficiency of the blockchain network and Cloudchain. Therefore, for future, we plan to consider multiple oracles and utilize more advanced techniques, such as reinforcement learning, to enforce truthfulness among these oracles and select the most reliable and efficient one in each round of verification.

REFERENCES

[1] M. M. Hassan, A. Alelaiwi, and A. Alamri, "A dynamic and efficient coalition formation game in cloud federation for multimedia applications," in *Proc. Int. Conf. Grid Comput.*, 2015, Art. no. 71.

[2] B. Ray, A. Saha, S. Khatua, and S. Roy, "Quality and profit assured trusted cloud federation formation: Game theory based approach," *IEEE Tran. Services Comput.*, 2018, early access, doi: 10.1109/TSC.2018.2833854.

[3] A. K. Bairagi, M. G. R. Alam, A. Talukder, T. H. Nguyen, D. E. Lee, and C. S. Hong, "An overlapping coalition formation approach to maximize payoffs in cloud computing environment," in *Proc. Int. Conf. Inf. Netw.*, 2016, pp. 324–329.

[4] C. A. Lee, "Cloud federation management and beyond: Requirements, relevant standards, and gaps," *IEEE Cloud Comput.*, vol. 3, no. 1, pp. 42–49, Jan./Feb. 2016.

[5] M. Taghavi, J. Bentahar, H. Otrok, and K. Bakhtiyari, "Cloudchain: A blockchain-based coopetition differential game model for cloud computing," in *Proc. Int. Conf. Service-Oriented Comput.*, 2018, pp. 146–161.

[6] R. Pal and P. Hui, "Economic models for cloud service markets: Pricing and capacity planning," *Theoretical Comput. Sci.*, vol. 496, pp. 113–124, 2013.

[7] M. Hadji, W. Louati, and D. Zeghlache, "Constrained pricing for cloud resource allocation," in *Proc. IEEE 10th Int. Symp. Netw. Comput. Appl.*, 2011, pp. 359–365.

[8] B. Xu, T. Qin, G. Qiu, and T.-Y. Liu, "Optimal pricing for the competitive and evolutionary cloud market" in *Proc. 24th Int. Conf. Artif. Intell.*, 2015, pp. 139–145.

[9] M. Fan, S. Kumar, and A. B. Whinston, "Short-term and long-term competition between providers of shrink-wrap software and software as a service," *Eur. J. Operational Res.*, vol. 196, no. 2, pp. 661–671, 2009.

[10] D. Niyato, A. V. Vasilakos, and Z. Kun, "Resource and revenue sharing with coalition formation of cloud providers: Game theoretic approach," in *Proc. 11th IEEE/ACM Int. Symp. Cluster Cloud Grid Comput.*, 2011, pp. 215–224.

[11] J. P. R. Coronado and J. Altmann, "Model for incentivizing cloud service federation," in *Proc. Int. Conf. Economics Grids Clouds Syst. Services*, 2017, pp. 233–246.

[12] O. A. Wahab, J. Bentahar, H. Otrok, and A. Mourad, "A stackelberg game for distributed formation of business-driven services communities," *Expert Syst. Appl.*, vol. 45, pp. 359–372, 2016.

[13] E. K. Asl, J. Bentahar, H. Otrok, and R. Mizouni, "Efficient community formation for web services," *IEEE Trans. Services Comput.*, vol. 8, no. 4, pp. 586–600, Jul./Aug. 2015.

[14] L. Zhao, L. Lu, Z. Jin, and C. Yu, "Online virtual machine placement for increasing cloud provider's revenue," *IEEE Trans. Services Comput.*, vol. 10, no. 2, pp. 273–285, Mar./Apr. 2017.

[15] J. Mendling et al., "Blockchains for business process management-challenges and opportunities," *ACM Trans. Manage. Inf. Syst.*, vol. 9, no. 1, 2018, Art. no. 4.

[16] D. Yermack, "Corporate governance and blockchains," *Rev. Finance*, vol. 21, no. 1, pp. 7–31, 2017.

[17] M. Klems, J. Eberhardt, S. Tai, S. Härtlein, S. Buchholz, and A. Tidjani, "Trustless intermediation in blockchain-based decentralized service marketplaces," in *Proc. Int. Conf. Service-Oriented Comput.*, 2017, pp. 731–739.

[18] Y. Jiao, P. Wang, D. Niyato, and Z. Xiong, "Social welfare maximization auction in edge computing resource allocation for mobile blockchain," in *Proc. IEEE Int. Conf. Commun.*, Kansas City, MO, 2018, pp. 1–6.

[19] S. Underwood, "Blockchain beyond bitcoin," *Commun. ACM*, vol. 59, no. 11, pp. 15–17, 2016.

[20] E. Münsing, J. Mather, and S. Moura, "Blockchains for decentralized optimization of energy resources in microgrid networks," in *Proc. IEEE Conf. Control Technol. Appl.*, 2017, pp. 2164–2171.

[21] Y. Zhang and J. Wen, "The IoT electric business model: Using blockchain technology for the Internet of Things," *Peer-to-Peer Netw. Appl.*, vol. 10, no. 4, pp. 983–994, 2017.

[22] C. Xu, K. Wang, and M. Guo, "Intelligent resource management in blockchain-based cloud datacenters," *IEEE Cloud Comput.*, vol. 4, no. 6, pp. 50–59, Nov./Dec. 2017.

[23] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, "Medrec: Using blockchain for medical data access and permission management," in *Proc. Int. Conf. Open Big Data*, 2016, pp. 25–30.

[24] X. Xu et al., "A taxonomy of blockchain-based systems for architecture design," in *Proc. IEEE Int. Conf. Softw. Architecture*, 2017, pp. 243–252.

[25] M. Taghavi, J. Bentahar, K. Bakhtiyari, and C. Hanachi, "New Insights towards developing recommender systems," *Comput. J.*, vol. 61, no. 3, pp. 319–348, 2017.

[26] L. M. Hocking, *Optimal Control: An Introduction to the Theory with Applications*. Oxford, United Kingdom: Oxford Univ. Press, 1991.

[27] D. Kraft, "Difficulty control for blockchain-based consensus systems," *Peer-to-Peer Netw. Appl.*, vol. 9, no. 2, pp. 397–413, 2016.

[28] M. Taghavi, J. Bentahar, H. Otrok, O. A. Wahab, and A. Mourad, "On the effects of user ratings on the profitability of cloud services," in *Proc. IEEE Int. Conf. Web Services*, 2017, pp. 1–8.

[29] M. Taghavi, J. Bentahar, and H. Otrok, "Two-stage game theoretical framework for iaas market share dynamics," *Future Gener. Comput. Syst.*, vol. 102, pp. 173–189, 2020.

[30] X. He, A. Prasad, S. P. Sethi, and G. J. Gutierrez, "A survey of stackelberg differential game models in supply and marketing channels," *J. Syst. Sci. Syst. Eng.*, vol. 16, no. 4, pp. 385–413, 2007.

[31] B. Robinson and C. Lakhani, "Dynamic price models for new-product planning," *Manage. Sci.*, vol. 21, no. 10, pp. 1113–1122, 1975.

[32] G. J. Gutierrez and X. He, "Life-cycle channel coordination issues in launching an innovative durable product," *Production Operations Manage.*, vol. 20, no. 2, pp. 268–279, 2011.

[33] J. Eliashberg and A. P. Jeuland, "The impact of competitive entry in a developing market upon dynamic pricing strategies," *Marketing Sci.*, vol. 5, no. 1, pp. 20–36, 1986.

Mona Taghavi is working toward the PhD degree at Institute of Information System Engineering, Concordia University, Canada. She was awarded the prestigious Canadian federal *Vanier Research Graduate* scholarship in 2016 and ranked first for FRQNT scholarship in Quebec. Her research interests include services computing, blockchain, game theory, machine learning, and recommender systems.

Jamal Bentahar received the PhD degree in computer science and software engineering from Laval University, Canada, in 2005. He is a professor with Concordia Institute for Information System Engineering, Concordia University, Canada and NSERC co-chair of the Discovery Grant Evaluation Group for Computer Science (2016-2018). His research interests include services computing, game theory, model checking, and multi-agent systems.

Hadi Otrok received the PhD degree in ECE from Concordia University, Canada. He is an associate professor with the Department of EECS at Khalifa University, UAE. He is an associate editor of the Ad hoc Networks (Elsevier) and a co-chair of several committees at various IEEE conferences. His research interests include IOT, services computing, network security, VANET, game theory, and mechanism design.

Kaveh Bakhtiyari received the Master of Artificial Intelligence degree from UKM, Malaysia. He is currently working toward the PhD degree in system engineering and computer science at the University of Duisburg-Essen, Germany, and The National University of Malaysia (UKM). His main research interests include recommender systems, reinforcement learning, and affective computing. He has been awarded DAAD Doctoral Scholarship in Germany, UKM Research Fellowship.