Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Feed4Cloud: Towards trustworthy QoE-aware cloud service monitoring using blockchain

Ioanna Angeliki Kapetanidou [a,b,*], Christos-Alexandros Sarros [c], Giannis Ledakis [c], Vassilis Tsaoussidis [a,b]

[a] ATHENA Research & Innovation Center, Xanthi, Greece
[b] Department of Electrical & Computer Engineering, Democritus University of Thrace, Xanthi, Greece
[c] UBITECH Ltd, Athens, Greece

ARTICLE INFO

Keywords:
Cloud computing
Blockchain
Quality of Experience (QoE)
Reputation
User feedback

ABSTRACT

The recent prevalence of microservice-based applications that leverage the capabilities offered by cloud and edge computing, has given rise to highly complex services which create new challenges for efficient monitoring and orchestration. In today's cloud environments, service monitoring is typically premised on technical Quality of Service (QoS) performance metrics, rather than on Quality of Experience (QoE) as perceived by users. In this paper, we posit that user feedback should also play a significant role in cloud service monitoring. However, we explicitly set a prerequisite: the trustworthiness of user feedback should not be considered guaranteed. Therefore, we have developed Feed4Cloud, the first system to complement QoS monitoring with exclusively trustworthy user feedback for QoE-aware cloud service management. The novelty of our solution lies in two key aspects: First, the establishment of an intermediate verification layer that validates user feedback before it is injected into the orchestration engine. The second key aspect is the use of Blockchain in this layer, as a means to record user feedback in a decentralized and secure way, aiming to achieve non-repudiation and ensure its integrity. In this paper, we present the architectural details of the Feed4Cloud prototype, while placing a particular focus on aspects regarding trustworthy evaluation of service performance. Furthermore, we provide evaluation results that validate the effectiveness of the introduced verification layer and demonstrate that QoE-based service evaluation can consistently be conducted in a trustworthy manner across a wide range of system conditions and user behaviors.

1. Introduction

1.1. Motivation

In today's highly distributed cloud and edge computing environments, there is a surge in the popularity of microservice-based applications, which however call for advanced monitoring and orchestration solutions. In common cloud settings, services are expected to meet the agreed-upon Quality of Service (QoS) requirements that correspond to specific Service Level Objectives (SLOs). Those SLOs are defined at associated Service Level Agreements (SLAs), i.e. technical services' performance contracts between the service provider and the client that set forth the terms of the transaction between the two parties. To be able to meet the stipulatory requirements, service providers need to monitor and configure their cloud services in a cost-efficient way, which, however, allows for providing a high user Quality of Experience (QoE). Typically, monitoring the performance and SLO compliance of cloud services is based solely on objective QoS measurements (e.g. response time, packet losses, jitter, etc.) obtained by relevant monitoring tools [1]. Consequently, important subjective aspects related to user experience, are neglected. Hence, the effectiveness of most monitoring tools is subject to an inherent lack of QoE-related information since the measured QoS does not always accurately reflect the users' perceived QoE. This stands in stark contrast to the ITU-T series of standards, namely the ITU-T Recommendations (ITU-T Recs), including ITU-T P.10 [2], which defines QoE as *"the overall acceptability of an application or service, as perceived subjectively by the end user"* and, thus explicitly differentiates it from QoS.

Therefore, considering that user QoE might be influenced by non-technical factors (e.g. cost, personal preferences, etc.), allowing users to communicate their QoE perception becomes essential to capture the

* Corresponding author at: ATHENA Research & Innovation Center, Xanthi, Greece.
E-mail addresses: ikapetan@athenarc.gr (I.A. Kapetanidou), asarros@ubitech.eu (C.-A. Sarros), gledakis@ubitech.eu (G. Ledakis), vtsaousi@ee.duth.gr (V. Tsaoussidis).

https://doi.org/10.1016/j.future.2024.107532
Received 24 November 2023; Received in revised form 10 September 2024; Accepted 12 September 2024

actual service performance better. Despite the growing importance of incorporating QoE monitoring into cloud service monitoring – particularly with the rise of advanced, often interactive applications – little effort has been made in that direction [3]. Although certain prior works have explored user-provided feedback as a means to raise QoE awareness in cloud services performance monitoring, they accept user feedback regardless of its trustworthiness (e.g. as is NORMA [4]) or propose solutions wherein the user feedback is collected and stored in a centralized way, thus introducing single-points-of-failure (e.g. [5,6]). Recent studies also highlight the need for novel mechanisms to assess the accuracy of opinion-based user feedback to ensure enhanced cloud service delivery, since this remains an open research issue [7].

1.2. Contributions

Feed4Cloud aims to fill this gap in cloud service monitoring by seamlessly incorporating user-generated feedback as an integral part of the monitoring plane to improve service orchestration. For instance, this can be achieved by using the QoE information for detecting SLA breaches or initiating QoE-driven elasticity actions. Needless to say, QoE insights aim to serve as an additional monitoring aspect to reinforce service monitoring rather than replace conventional QoS.

In this light, Feed4Cloud advances beyond the state-of-the-art in two distinct ways: First of all, by employing well-suited models to validate the provided feedback and allow only for trustworthy ratings to be embedded into the orchestration loop. To realize this, we propose an intermediate verification layer, as illustrated in Fig. 1, that evaluates the trustworthiness of the QoE information and filters out any suspicious data, instead of directly feeding the orchestrator with the raw feedback. The verification process is achieved by using pertinent algorithms to correlate the measured QoS with the expected QoE and compare it against the submitted QoE feedback, aiming to identify unjustifiable conflicts. Building upon the verification result, Feed4Cloud utilizes a reputation model to assess the credibility of the feedback-providing users and the quality of the cloud services. To further consolidate our system in this regard, our solution ensures that user feedback is collected in a transparent manner while ensuring its auditability. To this end, Feed4Cloud leverages Blockchain as a key enabling technology to record user feedback. This is the second innovative aspect of our approach. We claim that the benefits of using Blockchain for the overall system are straightforward: Firstly, it innately guarantees the immutability of the feedback history, thus facilitating holding users accountable for their feedback since they can neither alter their ratings to manipulate the evaluation system nor question the ratings' authenticity. Secondly, we ensure that the QoE feedback is auditable since it is timestamped and traceable; this allows the reputation management system to consider only ratings that can be linked with verifiable QoE in the reputation evaluation. Lastly, the recorded feedback can be used by different models to perform QoE verification and service quality evaluation in a decentralized way, while allowing reasoning about the models' results. This means that it is feasible for each evaluation service to employ a different evaluation model but still, its output can be validated based on the shared collected ratings. In general, storing the user feedback in a distributed ledger ensures certain properties (integrity, non-repudiation, etc.) even if the blockchain nodes to which the feedback is sent are operated by different parties. Overall, Feed4Cloud seeks to bring together user feedback trustworthiness, reputation management and Blockchain technology, aiming at advanced QoE-aware cloud service management. To the best of our knowledge, Feed4Cloud is the first solution to incorporate user-generated, yet evidently trustworthy QoE feedback to complement QoS information in cloud service monitoring. These enhanced monitoring capabilities can improve the management and orchestration of distributed microservice-based applications, enhancing the orchestration engine's feedback loop with user-provided information. This can potentially improve the orchestrator's deployment decisions such as (micro-)service placement, scaling, etc. In this context, the design and prototype Feed4Cloud implementation are thoroughly presented. In addition, a comprehensive evaluation of the reputation-based verification layer is provided, considering both normal and degraded QoE service performance, as well as different honest and dishonest user practices.

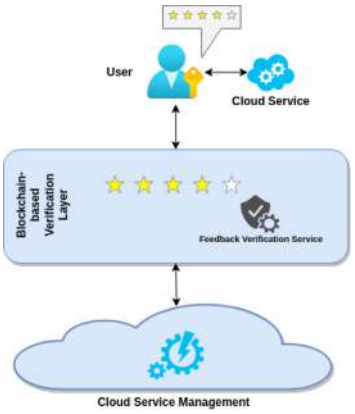


Fig. 1. Feed4Cloud conceptual layers.

1.3. Paper organization

The rest of the paper is organized as follows: Section 2 provides an overview of the state-of-the-art of user feedback-aware and blockchain-assisted cloud monitoring. In Sections 3 and 4, we delineate the design primitives and the Feed4Cloud main components, respectively, while in Section 5 we delve into the implementation details. Then, in Section 6 we present our real-world deployment, along with extensive evaluation results of the intermediate layer mechanisms. Finally, in Section 7 we discuss Feed4Cloud's reputation system security specifics and propose potential real-world use cases and future extensions.

2. Related work

In this section, we compare our proposed Feed4Cloud system with related scientific works in terms of our two main novelty pillars, i.e. the incorporation of verified user QoE feedback in the cloud monitoring process and leveraging blockchain to assist trustworthy QoE monitoring.

2.1. User QoE-related feedback in cloud monitoring

The integration of QoE-related information has been investigated in various application domains in the Cloud-to-Edge continuum, ranging from cloud- or edge-hosted multimedia [8,9] or gaming services [10] to Internet of Things (IoT)-based clouds [5,11], or even in the context of the 5G networks [12,13].

This topic is also being actively investigated by the Qualinet Working Group 1 Task Force on "Managing Web and Cloud QoE" [14], dedicated to bringing QoE closer to practical applications by providing workable QoE models to be integrated into cross-layer network/application management schemes, Web applications and cloud service provisioning.

In this context, QoS-to-QoE correlation models have been developed to accurately assess user experience based on service performance, the most notable [15] being the exponential interdependency of QoE and QoS [16–18], also known as IQX hypothesis, and the logarithmic relationship [19], as recommended by ITU-T. Although QoS to QoE mapping has been used in related works for cloud service evaluation [4,8,12], only [8] specifies the mapping function being used,

i.e. ITU-T P.1203. Still, as of now QoS to QoE mapping has been leveraged as a means to automatically infer user experience. In contrary, in Feed4Cloud, we take a human-in-the-loop approach because user-provided feedback can capture user experience more accurately and correlate QoS to QoE aiming to confirm their reasonability. To this end, we choose to employ the exponential IQX hypothesis because it has been found to be superior to the logarithmic mapping [15].

However, the majority of related works does not consider mapping QoS to QoE. Instead, they mostly prefer using QoE input as a completely independent criterion [5,6,9,13,20–29]. In some cases [6,9,21,22,24,27–30] even metrics totally irrelevant from measurable QoS are aggregated into the evaluation model. For instance, [21] assesses user experience based on document readability, support satisfaction and operability and combines the result with technical factors such as CPU utilization and network latency. Similarly, [28] considers metrics such as availability and price. Although such metrics offer further insights, they normally are influenced by individual user requirements and preferences [9,20,21,24,25], thus being objective and difficult to validate. In contrast, [11] relies solely on QoS-related feedback, thus requiring methods such as feedback similarity to ensure its reliability. However such methods are innately prone to shilling attacks[1] [31,32]. Among the user feedback-based related solutions, several factor such information into the service evaluation without even employing any measures to assess its validity [13,20] or without considering the user credibility throughout such process [6,13,20]. Conversely, those two constitute essential features of Feed4Cloud as they increase the overall evaluation reliability [5]. User feedback trustworthiness evaluation in the cloud has been thoroughly studied by the research community. Related works normally use mathematical models to detect fake feedback. The most widely used methods involve computing the similarity between the feedback provided by a specific user with the feedback of other users [6,11,20,25], computing weighted scores to minimize the impact of inconsistent rating behaviors [26,27], using fuzzy logic to incorporate user opinions into the service evaluation [21,29] or calculating the distance between subjective user feedback from objective QoS assessments to determine user feedback trustworthiness [5,33].

Therefore, although prior works have already highlighted the importance of user feedback in service evaluation, none considers user experience in such a way that it can be reliably verified, i.e. by correlating it with objective, measurable criteria. Feed4Cloud endeavors to rigorously bind it to monitored QoS through well-established correlation models to inject subjective, yet unquestionable insights into this operation.

Using feedback verification results to compute users' credibility and determine the impact of their feedback in the reputation score evaluation accordingly, constitutes another commonly used methodology [5,11,21,25,27,29]. Such approaches lack any measures to protect users' privacy though. Nonetheless, CloudArmor [28] argues that user feedback credibility should be evaluated without breaching the users' privacy. To achieve this, CloudArmor employs a Zero-knowledge credibility proof protocol (ZKC2P) premised on anonymized information. In Feed4Cloud, we also embrace privacy as a requirement but rely on blockchain-based user registration to preserve user anonymity.

Overall, to date, QoE specifics have been mainly considered as an auxiliary criterion to assist cloud users in the service selection process [5,6,20–22,30,33], but also to allow for policy adaptation in runtime in cloud video streaming [8,9], as well as, gaming services [10]. From service providers' perspective, user-generated QoE insights are also particularly important as they enable improved cloud service orchestration effectiveness. By closely monitoring user-perceived QoE, service providers can detect potential quality degradations and react on time to avoid user churn, while they might also rely on QoE measurements to achieve a resource-efficient service deployment at the minimum operational cost [34,35]. Especially in 5G networks, QoE-aware orchestration constitutes a recent research trend. In this regard, NORMA [4] considers QoS measurements mapped to QoE scores that feed a designated service and network resource orchestrator to allow for informed decisions. In a similar fashion, in [12] enhanced orchestration is accomplished through a QoE-aware elasticity policy that dynamically adapts to the traffic conditions aiming to maintain high QoE, while in [13] dynamic resource allocation in fog settings is improved via a QoE-based mechanism particularly designed for Tactile Industrial IoT (IIoT) applications. Indeed, certain works propose defining the user requirements regarding service quality, either by explicitly specifying an agreed-upon trust level as an SLO [23] or by incorporating user feedback directly in the SLA monitoring process [24,36].

All in all, Feed4Cloud differentiates from the aforementioned works in the following ways: (i) some works either do not assess feedback trustworthiness and/or user credibility; two vital Feed4Cloud features, (ii) Feed4Cloud is the only solution leveraging QoS to QoE mapping to evaluate feedback trustworthiness, (iii) privacy is an aspect neglected in previous works, except for [28], but unlike the latter, Feed4Cloud takes a blockchain-based authentication approach to ensure user privacy, and (iv) all those works rely on central trusted network parties or cloud providers for QoE monitoring, as well as for collecting and storing the provided feedback in associated locally-managed databases. This renders them inherently susceptible to tampering and does not provide any precautions on user accountability.

2.2. Blockchain-assisted cloud service monitoring

Unlike previous centralized approaches, a first attempt to provide decentralized, reputation-aware SLA monitoring has been done in Peer-to-Peer (P2P) networks [37]. Service reputation is evaluated based on QoS performance. Each peer utilizes personalized models to evaluate and report the QoS. Honest client behavior ensures that higher QoS results in higher reported trust values, motivating providers to provide high QoS to increase their revenue. The reputation system is open such that each participant can become aware of another peer's evaluation of the provider's reputation, but also identify clients reporting false evaluations. Still, this solution lacks any mechanism to assess user credibility in an automated way and consequently, take this into account in service evaluation.

To further contribute towards SLA monitoring decentralization, [38] introduces the notion of 'SLA scores' which denote the conformance with the SLA conditions. SLA scores are computed by relying on objective network measurements, which are verified using smart contracts. Therefore, Blockchain technology is used to automatically detect and penalize cheating parties that provide false testimonies. Blockchain is also used for recording network measurements and SLA scores. However, SLA scores are premised solely on monitored technical metrics, thus neglecting user experience, let alone user feedback, the core Feed4Cloud asset.

Along these lines, a more recent research direction suggests that SLO violations are detected either by the involved parties directly [39,40], or by delegated network members, that are in turn rewarded by the Blockchain [41,42].

In [39] dedicated smart contracts are used to automatically assess the credibility of fog IoT clients in a public Ethereum-based setting. The reputation of the service providers is also calculated through associated smart contracts by taking into account the credibility of the feedback provider. To motivate clients to provide trustworthy ratings, they are asked to transfer a deposit, returnable once they have proven to be reliable. However, inferring the reputation scores based entirely on the user-generated feedback, with no guarantee that the provided

[1] In shilling attacks, the attacker targets the similarity-based system; it creates fake profiles that strongly correlate with victims' profiles in order to manipulate their recommendations' lists.

feedback is aligned with the actual QoS, makes this system vulnerable to Collusion or Sybil Attacks [43].

To cope with this drawback, in [40], besides the service providers and clients, a monitoring authority is registered to the blockchain. The latter monitors the compliance of the service endpoints with the agreed SLOs and reports any detected violations. A private blockchain is used to create an immutable history of records to facilitate the monitoring of SLA enforcement. In an alternative approach, a new blockchain-registered entity is introduced, namely the "witness", which is responsible for detecting and reporting service violations [41]. Independent witnesses are selected randomly per transaction, out of a decentralized witness pool. To incentivize blockchain users to participate as witnesses, a payoff function is proposed so that they gain revenue for offering reporting services. Building upon the witness model, [42] utilizes a permissioned blockchain formed by a federation of randomly selected oracles.[2] SLAs are transformed into smart contracts fed by the oracles' objective monitoring data, enabling distributed SLA enforcement. BlockQoS [45] leverages blockchain to enable users to request on-demand sessions video conferencing sessions with prespecified QoS guarantees in a traceable manner, to monitor SLA compliance and also to automate monetization.

Recently, a tri-chain blockchain model has been proposed to facilitate service selection in cloud manufacturing [46]. Service providers offer resources, service consumers submit manufacturing tasks and solvers with the capacity to meet associated hardware requirements execute those tasks and upload execution results in a solution pool. The solver of the solution unanimously selected by the provider and the consumer publishes a respective contract, the service delivery information but also updated credibility values of all involved parties to be securely and permanently stored in the blockchain.

In the context of multi-cloud systems, blockchain networks have been used to store logs of SLA violations and render them visible to potential service users [47].

As obvious, currently, blockchain-based reputation has been investigated for QoS monitoring. Therefore, Feed4Cloud is the first solution fostering the use of blockchain for QoE-aware cloud service evaluation.

2.3. Beyond state-of-the-art

Table 1 provides an overview of the salient features of the presented works. The *'QoS/QoE'* column indicates whether the respective approaches use QoS-QoE correlation models (*'mapping'*), combine distinct objective and subjective metrics (*'combined'*) or rely solely on QoS information (*'QoS'*).

Based on the literature review, to the best of our knowledge, Feed4Cloud emerges as the first coherent system that supports distributed user feedback collection and recording, QoE verification via pertinent correlation models, user credibility assessment and service evaluation to enable enhanced cloud service management, all at the same time.

3. Feed4Cloud design

The key objective of Feed4Cloud is to enhance cloud service monitoring and orchestration by enabling additional insights related to QoE aspects, complementing the default QoS metrics. To achieve this, Feed4Cloud depends heavily on user-provided feedback which reflects the service evaluation in terms of the user-perceived QoE. However, we acknowledge that incorporating user feedback might open the door to manipulative behaviors. For instance, rogue raters might attempt to exploit the feedback collection system to intentionally lead to a falsely lower service reputation score. The latter, could, in turn, result in faulty orchestration decisions.

Considering this, it becomes self-evident that shielding the feedback system is vital. To this end, we set two primary design goals for our solution: The first one is to collect user feedback in a secure and transparent manner, whilst the second is to ensure the trustworthiness of the obtained feedback. Feed4Cloud safeguards the feedback loop by collecting feedback only from authenticated parties, storing it in a tamper-proof manner and also validating feedback prior to forwarding it to the monitoring engine. Hence, we go one step further and consolidate our solution by establishing procedures that guarantee service evaluation is based on valid QoE information.

In order to achieve our design goals, we leverage Blockchain technology for securely collecting and storing the user ratings but also rely on well-established models to validate user feedback. More specifically, users are required to sign in using their digital wallet and authenticate themselves in order to gain permission to submit their ratings to restrict unauthorized access. Moreover, to allow for transparency and rating provenance during feedback collection, we stipulate that feedback is uploaded to a distributed ledger as soon as it is submitted. That way the stored ratings are not locally controlled by a single entity, but are rather fully traceable and can be easily audited ex post facto, while they remain tamper-proof and their integrity cannot be questioned by any system user. Furthermore, recording ratings and mapping them to the associated rater's on-chain also caters to the non-repudiation of raters' actions since they cannot alter or dispute previous ratings. This is very important as it facilitates establishing a concrete credibility mechanism to ensure the accountability of users; an aspect considered crucial in reputation systems. In this context, in Feed4Cloud, we provision an oracle smart contract [48,49] that interacts with the external oracles[2] (i.e. the service users) to gather QoE feedback.

Moreover, given that recorded ratings are timestamped and traceable, they can be verified in a decentralized manner at any time. In other words, the oracle smart contract can be interconnected either with other smart contracts that are delegated with the computation of the service reputation and/or with off-chain evaluation mechanisms. In any case, the trustworthiness of the oracles-raters is not inherently guaranteed [48], necessitating additional mechanisms to evaluate oracle-provided input [7].

In this scope, Feed4Cloud implements designated external off-chain mechanisms that are fed by the oracle smart contract. They leverage well-suited models and methodologies to subsequently validate submitted feedback, detect and filter out fake ones. They also employ bespoke algorithms to assess two fundamental metrics of our system, namely the *user credibility* and the *service reputation score*. The user credibility is calculated based on the trustworthiness of the user feedback and indicates whether the specific rater can be trusted. The service reputation score serves as an indicator of the service quality in terms of QoE performance. The aforementioned evaluations are provided by the RateTrust Feedback Service (RTFS).

The trustworthy user feedback, and indeed the computed service reputation score, is being integrated into the monitoring plane aiming to enable improved service configuration and advanced cloud-to-edge orchestration.

The Feed4Cloud conceptual design is depicted in Fig. 2, wherein the solid and dashed lines represent information transfers and interactions between the involved entities, respectively.

4. Feed4Cloud architecture

The Feed4Cloud architecture depicted in the above component diagram (Fig. 3), consists of the following components:

- A widget for user ratings collection
- A monitoring system for QoS data collection
- A service orchestrator

[2] In the context of blockchain, the term "oracle" refers to an entity that feeds a smart contract with external data [44].

Table 1
Related works overview.

Reference No.	QoS/QoE	User-provided feedback	Fake feedback detection	User credibility assessment	Monitoring/ Orchestration	Blockchain
[4]	Mapping				✓	
[5]	Combined	✓	✓	✓		
[6]	Combined	✓	✓			
[8]	Mapping		✓		✓	
[9]	Combined	✓	✓	✓	✓	
[11]	QoS	✓	✓	✓		
[12]	Mapping				✓	
[13]	Combined	✓			✓	
[20]	Combined	✓	✓	✓	✓	
[21]	Combined	✓	✓	✓	✓	
[22]	Combined	✓	✓	✓		
[23]	Combined	✓			✓	
[24]	Combined			✓	✓	
[25]	Combined	✓	✓	✓	✓	
[26]	Combined	✓	✓	✓		
[27]	Combined	✓	✓	✓		
[28]	Combined	✓	✓	✓		
[29]	Combined	✓	✓	✓		
[33]	Combined	✓				
[37]	QoS	✓	✓		✓	✓
[38]	QoS		✓	✓	✓	✓
[39]	QoS	✓		✓	✓	✓
[40]	QoS				✓	✓
[41]	QoS		✓		✓	✓
[42]	QoS				✓	✓
[45]	QoS				✓	✓
[46]	QoS			✓	✓	✓
[47]	QoS				✓	✓
Feed4Cloud	Mapping	✓	✓	✓	✓	✓

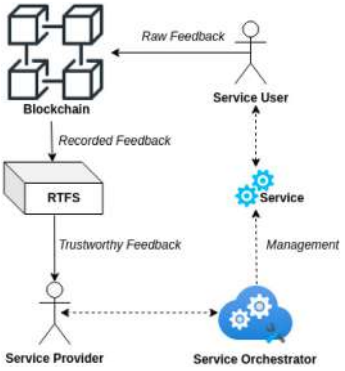


Fig. 2. Feed4Cloud concept.

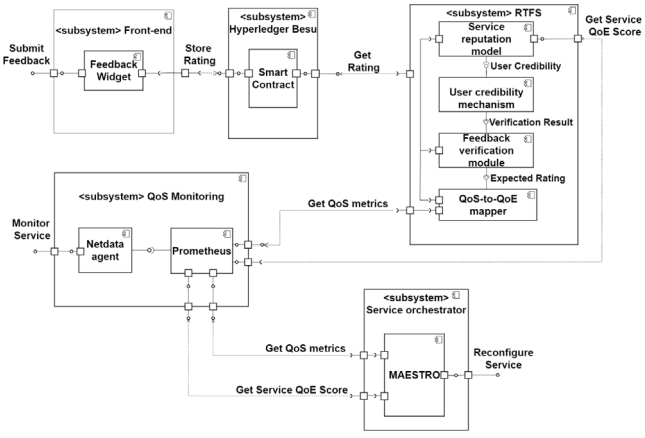


Fig. 3. Feed4Cloud architecture.

- A blockchain-hosted smart contract
- A QoS-to-QoE mapper
- A Feedback Verification module
- A Service Reputation model
- A User Credibility mechanism

The last four components comprise the RTFS.

The overall architectural components interact with each other in the following way: First of all, we assume that a service has been deployed using the service orchestrator and the user has previously utilized the service. This user provides some feedback for the service, based on its Quality of Experience. The feedback will be provided using the Feed4Cloud rating collection widget. The front-end widget communicates with the oracle smart contract to send the raw user ratings to be securely stored on the blockchain network. Those – immutable – ratings will then be available in the distributed ledger for anyone participating in the network, to verify their authenticity and integrity.

On the backend, the Feedback Verification module also interacts with the oracle smart contract to retrieve the user ratings from the blockchain. It also retrieves the QoS measurements for the deployed service from the monitoring data collection module. The two inputs (QoE and QoS) are then compared using a correlation model and ratings will be characterized as either trustworthy or potentially malicious. Based on that information, the credibility mechanism infers the user credibility scores. A reputation model also assigns the service

reputation. The potency of a particular user's feedback in the service reputation score is determined by the user credibility, while only trustworthy QoE feedback will be considered in the service evaluation. The resulting service reputation score (calculated based on the filtered QoE) is eventually fed into the service orchestrator.

The service orchestrator receives the filtered QoE information and uses it to supplement the information retrieved from its QoS monitoring plane. Based on both information sources, it monitors service performance and can (re-)configure services to ensure SLO compliance.

5. Implementation

In this Section, we lay out the implementation details of the individual Feed4Cloud components and their integration.

5.1. Feedback collection widget

Our Feedback Collection Widget serves as the Feed4Cloud front-end for the services' users. After a user's interaction with the web service has been completed, the service user can use the widget to provide feedback on its perceived QoE.

This component is implemented as a browser-based widget, using *vue.js* [50]. The service provider integrates the rating widget in the front-end code of its application, using the provided instructions and code artifacts. In our prototype implementation, the rating widget has been integrated with the service using Node.js v16.13.1. The user feedback is provided using a GUI, in the form of 1- to 5-star ratings.

The widget has also been integrated with MetaMask [51], which acts as a user's blockchain wallet and provides authentication services before a rating can be submitted. This way, we provide a first layer of defense that protects our reputation system from large-scale attacks by unauthorized users who submit fake ratings. After a user logs in and submits a rating for the service, the widget submits the rating as a transaction in the blockchain using the *web3.js* [52] library.

5.2. Monitoring system

This system is responsible for gathering monitoring information about the deployed services. It is implemented using Netdata 1.38 [53], Prometheus 2.43.0 [54] and OpenTelemetry [55] (opentelemetry-exporter-prometheus V0.38.0).

The Netdata agent is used to collect QoS information from services; it is installed on the nodes running the services to obtain the relevant metrics. OpenTelemetry is used to collect the verified QoE feedback; it instruments RTFS and exports the custom *'service QoE score'* metric (i.e. the score calculated based on the filtered QoE) to Prometheus. Prometheus acts as the monitoring back-end; it collects the QoS and QoE metrics from the aforementioned components and stores them as time-series data that can be processed and aggregated. It then makes those metrics available to the service orchestrator, allowing for enhanced observability of cloud services' behavior.

5.3. Service Orchestrator

Feed4Cloud's Service Orchestrator is based on MAESTRO [56], a software solution provided by UBITECH. MAESTRO can deploy and manage the lifecycle of a cloud service's distributed components, using either VM- or container-based virtualization. Besides the service monitoring system, described in the previous subsection, the component includes a visual graph editor and a service elasticity management framework based on graphically-designed rules and policies.

In Feed4Cloud we have extended its monitoring plane by enabling the collection of trustworthy user feedback. This way, we have created the first (to the best of our knowledge) cloud service orchestrator that incorporates a human-in-the-loop approach and utilizes QoE metrics in its orchestration process.

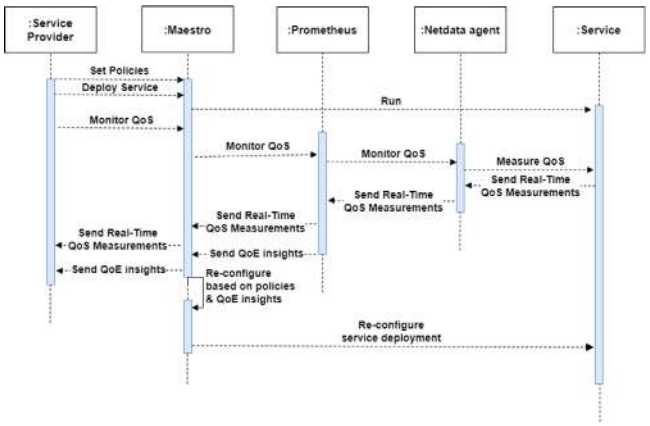


Fig. 4. Feed4Cloud service deployment & monitoring sequence diagram.

The following sequence diagram illustrates the order in which all necessary interactions between the involved components take place to deploy and monitor the QoS performance of the Feed4Cloud service, and indicate any need for re-configurations (see Fig. 4).

5.4. Smart contract

As already discussed, to enable distributed and immutable user feedback, in Feed4Cloud we have developed an oracle smart contract that interacts with the external oracles[2] (i.e. the service users/raters) to gather and securely record QoE ratings.

The implemented smart contract was developed using Solidity v0.8.18 [57]. In our smart contract, each user rating is represented by a struct, like so:

Listing 1: User Ratings

```
struct Rating {
        uint serviceId;
        address userId;
        uint rating;
        uint timestamp;
    }
```

wherein the *serviceId* represents a 6-digit id of the deployed service instance, the *userId* corresponds to the user's digital wallet hex address, the *rating* is the submitted rating and the *timestamp* denotes the date of the rating submission in a Unix timestamp format.

It is noteworthy that by utilizing the user's digital wallet hex address as the identifier, our approach can be considered pseudonymous as it conceals the individual's identity, thus safeguarding user privacy.

The smart contract currently supports the following seven functions:

1. setRating(uint serviceId, uint rating): Called on the front-end side to submit a rating
2. getRatings(): Returns all the recorded ratings
3. getUserRatings(address userId): Returns the ratings assigned by a specific user, if any
4. getLatestUserRating(address userId): Returns the latest rating submitted by a specific user that has been recorded, if any
5. getAverageUserRating(address userId, uint serviceId): Returns the average value of the ratings assigned by a specific user to a specific service, if any
6. getServiceRatings(uint serviceId): Returns the ratings assigned to a specific service, if any
7. getLatestServiceRating(uint serviceId): Returns the latest rating for a specific service that has been recorded, if any

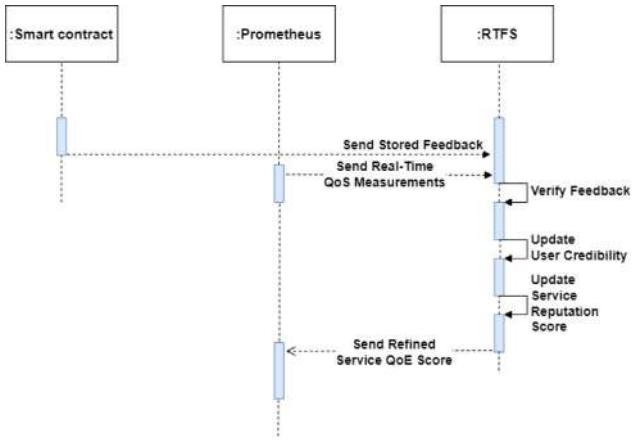


Fig. 5. RTFS inter-communication sequence diagram.

The smart contract has been tested locally on a Ganache network [58] using the Truffle suite [59]. Moreover, it has been deployed in a real-world setting; particularly on a Hyperledger Besu client node on the ALASTRIA Red B Network [60]. This is a public-permissioned Blockchain network consisting explicitly of accepted nodes. For this deployment, a personal API key was provided to us to make calls to the node's RPC API, allowing us to submit transactions and interact with the smart contract in a controlled environment.

5.5. RateTrust Feedback Service (RTFS)

RateTrust Feedback Service (RTFS) is a suite of interrelated Python scripts (developed using Python 3.9.16), each one corresponding to an RTFS sub-component, namely the:

- QoS-to-QoE mapper
- Feedback Verification module
- User Credibility mechanism
- Service Reputation model

Those components are designed to support the feedback trustworthiness verification, service evaluation and user credibility assessment. To this end, in the back end, RTFS executes a set of actions as presented in the below sequence diagram (see Fig. 6):

As depicted in Fig. 5, the series of actions taking place is the following:

1. First of all, RTFS obtains the QoS and QoE information.
   (a) RTFS fetches the real-time QoS measurements as captured by Prometheus.
   (b) RTFS maps the QoS metrics with user QoE and thus, infers the expected QoE.
   (c) At the same time, RTFS retrieves the QoE feedback recorded in the blockchain ledger.
2. Based on the information from the previous step, RTFS proceeds to feedback verification.
3. The filtered QoE (denoted as RefinedQoE in the diagram) is sent back to Prometheus.
4. The last two steps, i.e. the update of the user credibility value and the update of the service reputation score, correspond to the evaluation of the service users and the services, respectively.

A distinct RTFS sub-component is called to execute the internal RTFS operational steps, as shown in the following sequence diagram.

The implementation of the RTFS sub-components is described in detail below.

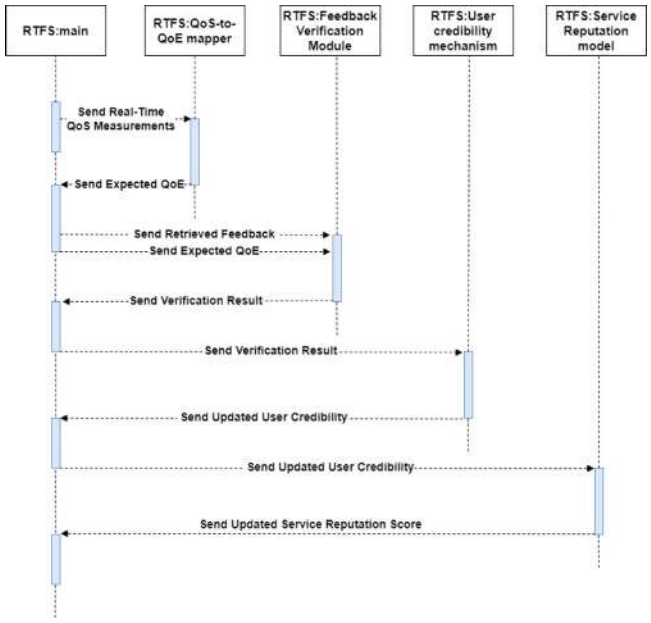


Fig. 6. Intra-RTFS sequence diagram.

5.5.1. QoS-to-QoE mapper

The *QoS-to-QoE mapper* is responsible for correlating the objective QoS measurements with a subjective QoE value.

To achieve so, the *QoS-to-QoE mapper* takes as input the monitored QoS metrics and thereafter, employs the IQX hypothesis [16–18]. The IQX hypothesis is a fundamental QoS-to-QoE mapping function based on the exponential interdependency of QoE and QoS:

$$QoE = \alpha \cdot e^{-\beta QoS} \tag{1}$$

In the above formula, α is set to 4 to reflect the 5-point rating scale [16] and β is a sensitivity parameter that has to be tuned for each particular QoS metric to accurately translate the actual QoS to a Mean Opinion Score (MOS) value, denoting user QoE. In our implementation, we have considered six QoS metrics, namely the: (i) *network throughput*, (ii) *packet drops*, (iii) *packet errors,* (iv) *idle jitter*, (v) *system uptime*, and (vi) *CPU usage*. These metrics are measured by the Netdata agent of the monitoring system.

By relying on the IQX model, *QoS-to-QoE mapper* infers the expected user-perceived QoE [61] for each QoS metric.

5.5.2. Feedback verification module

Feedback verification, a key functionality of RTFS, is performed in this module. The *Feedback Verification module* is fed by the expected QoE per QoS metric as computed by the *QoS-to-QoE mapper* and also the actual user rating as recorded in the Blockchain. Thereafter, it compares this rating against each and every expected QoE value and counts the number of mismatches. In our case, we consider that a rating is not valid if there are more than three mismatches (i.e. half the QoS metrics). The rationale behind this is the following: The users provide only a single rating denoting their overall experience. The user rating is expected to be in accordance with most of the actual QoS. Therefore, the rating is not taken into account if it is not aligned with the majority of the QoS measurements.

To identify any mismatches, this module will calculate either the numerical difference between two ratings or the euclidean distance between two triangular fuzzy numbers, depending on the reputation model being used (see Section 5.5.4 below for more details). In any case, the result reflects the deviation between the expected and the actual user rating.

The respective deviation is used to determine whether the submitted rating is reasonable (and thus, the user feedback is successfully verified) or if there is an unjustifiable deviation (and thus, the user feedback is considered an indication of potential user misbehavior). That way, the *Feedback Verification module* detects unreliable users that provide fake feedback and factors their feedback out of the service reputation calculation.

5.5.3. User Credibility mechanism

The *User Credibility mechanism* evaluates the credibility of users based on the trustworthiness of their provided feedback, as assessed by the *Feedback Verification module*. The mechanism firstly computes a $Credibility_{user_{new}}$ score which expresses the latest credibility score of the user and then uses this score internally to calculate the final user credibility based on both the current and historical user credibility assessments.

In the $Credibility_{user_{new}}$ calculation, aiming to provide counter-incentives for users providing fake feedback, we choose to rate each evaluator based on the "linear increase, exponential decrease" principle, like so:

$$Credibility_{user_{new}} = \begin{cases} 5, & \text{if feedback is verified} \\ -\theta e^{\alpha \cdot (1 - No.FakeRatings)}, & \text{otherwise} \end{cases} \quad (2)$$

where θ is a penalty factor representing the strength of the punishment and α is an adjustment factor. This principle originates from congestion control [62], but has also been applied to reputation models [63,64].

Based on this approach, the credibility increases gradually as long as a user provides truthful feedback, but it will be significantly decreased on fake feedback submissions. Adopting such a credibility model, allows us to control the impact that reporting fake feedback has on the overall credibility score and consequently, the effort that misbehaving users will need to recover from a penalty and converge back to regular ratings. Additionally, it provides protection against malicious users who occasionally submit verifiable feedback in an attempt to slip through the credibility mechanism.

Subsequently, to infer the credibility score of each user, both the current credibility $Credibility_{user_{new}}$ assigned to the particular user and the user's previous credibility score are considered, using an Exponential Weight Moving Average (EWMA), as follows:

$$Credibility_{user} = w * Credibility_{user_{old}} + (1 - w) * Credibility_{user_{new}} \quad (3)$$

where w denotes a weight factor. Provided that more recent user credibility scores capture more accurately the current user behavior, a higher weight[3] is given to the current credibility.

5.5.4. Service reputation model

Feed4Cloud is premised on a service reputation model assigning scores to each service based on its performance to, ultimately, improve cloud monitoring and orchestration. To rely on sound information for this purpose, only trustworthy feedback is considered in the service evaluation. Feed4Cloud currently supports two distinct well-known reputation models: a *Bayesian* and a *Fuzzy* one. Needless to say, other reputation models might be utilized as well to better fit the needs of respective implementation scenarios.

Bayesian model. For starters, we implemented a Bayesian reputation model. This is a widely adopted reputation calculation method that has been adopted by several popular rating systems, including the Trustpilot [65] online rating platform. To be precise, the Trustpilot computes a trust score for each stakeholder by accumulating user feedback and factoring it into the calculation formula based on three factors: time span (so that most fresh reviews hold a higher weight in the calculation), frequency (to ensure that the higher the feedback collection rate, the more stable the trust score), and Bayesian average (to treat fairly new evaluatees) [65]. Although the actual weights used in Trustpilot's trust formula are not publicly disclosed, we relied on the available information to implement a Bayesian reputation model that is aligned with Trustpilot's trust model.

To this end, similar to [65], we automatically include seven ratings and assume an initial reputation score equal to 3.5, upon initialization. Thereafter, the current rating ($Reputation_{service_{new}}$) is computed using the Bayesian formula:

$$Reputation_{service_{new}} = \frac{C * prior + \sum ratings}{C + \#ratings} \quad (4)$$

Wherein *prior* is the average rating we expect given a number of observations, C. We replace *prior* with the last calculated reputation score and C with the number of ratings ($\#ratings$), recorded so far.

The resulting $Reputation_{service_{new}}$ is factored into an EWMA to infer the updated reputation score, like so:

$$Reputation_{service} = |z - 0.8| * Reputation_{service_{old}} + |z - 0.2| * Reputation_{service_{new}} \quad (5)$$

Aiming to minimize the impact of feedback provided by users who have been dishonest, we set the weight z equal to $\frac{\text{rater's credibility}}{5}$ to determine its impact on service evaluation. That way, our solution requires that the less trustworthy a rater is, the less its feedback influences the service's reputation score.

Moreover, it is important to note that by fine-tuning θ in formula (2) and w in (3), slanders are quickly detected and punished, hence leaving no prospect for attackers to re-join as newcomers to increase their impact on service evaluation and eventually, mitigating the cold-start problem.

Fuzzy model. Aiming to provide a user-centric design, RTFS also employs fuzzy logic. Fuzzy logic is a widely used method, adopted in a variety of applications [66], and especially in cloud computing reputation models [27,29,67–70]. Considering that users might provide their feedback using a numeric rating scale (e.g. on a 5-point Absolute Category Rating (ACR) scale) but might also express their opinion in a fuzzy fashion (e.g. evaluating fuzzy QoE metrics, such as pricing, overall satisfaction, etc, using terms like Poor/Good/Excellent), adopting a fuzzy logic-driven model becomes essential. Therefore, although our current feedback collection widget implementation is based on an ACR scale, the QoE verification and overall service evaluation should also be able to be handled by a fuzzy model that can support potentially heterogeneous input data.

To accommodate the fuzzy QoE input, we follow the modified Fuzzy Analytical Hierarchical Process (FAHP) as proposed in [21]. In this work, the authors construct a Relative Attribute Comparison Matrix (RACM), similar to the decision matrix constructed in the defacto FAHP, to compare the actual user rating (A_{user}) with an ideal rating (A_{ideal}) of a virtual user. We embrace their model but we only consider A_{user} if it is verified and we also use the expected QoE (as obtained by the *Qos-to-QoE mapper* 5.5.1) ($A_{expected}$) rating instead of the ideal one:

$$RACM = \begin{bmatrix} 1 & \frac{A_{user}}{A_{expected}} \\ \frac{A_{expected}}{A_{user}} & 1 \end{bmatrix} \quad (6)$$

Thereafter, we employ the defacto FAHP to infer the latest service score $Reputation_{service_{new}}$, so that $Reputation_{service_{new}} \in [0, 1]$. This $Reputation_{service_{new}}$ expresses the degree of its importance in the service evaluation. To normalize it, we then multiply it with the submitted rating (A_{user}), extending its range from 0 to 5.

Finally, the service evaluation process is again concluded using Eq. (5) to ensure that the impact of a user's feedback on the overall reputation score is determined by its credibility.

[3] For the experiments presented in this paper, it is set to 0.75.

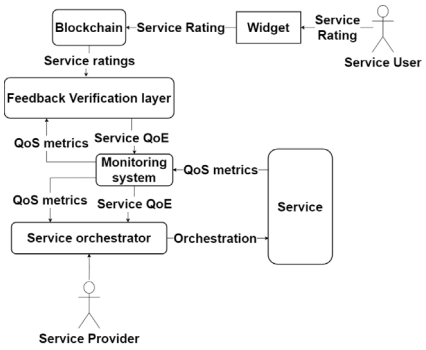


Fig. 7. Feed4Cloud information flow.

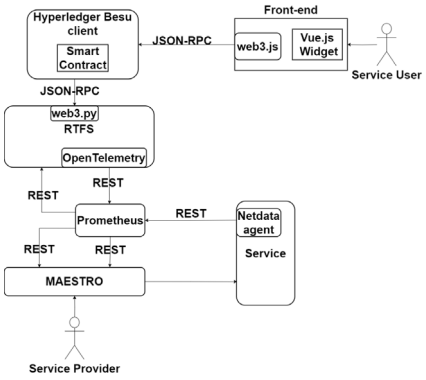


Fig. 8. Feed4Cloud inter-component communication.

5.6. End-to-end information flow

The end-to-end information dissemination within the system pipeline is depicted in Fig. 7. In more detail, the RTFS sub-system is triggered periodically by a scheduler. RTFS then communicates with the Hyperledger Besu client to find out whether there are any new ratings to be verified. In turn, RTFS periodically sends the service reputation score to the orchestrator, regardless of whether it has been changed.

To support the end-to-end information flow, the interfaces interconnecting the Feed4Cloud components have been developed using suitable communication protocols (Fig. 8).

The Feedback Collection Widget submits the user ratings as transactions to the blockchain, communicating with the dedicated smart contract running on the Hyperledger Besu client. This is achieved using the web3.js library, which uses JSON-RPC calls to invoke the functions exposed by the smart contract's Application Binary Interface (ABI). The connection to the blockchain is mediated by MetaMask, which acts as a gateway to the network in which the smart contract is deployed.

Focusing on the back end, RTFS retrieves the most recent user ratings from the blockchain using the web3py library [71] to communicate with the Hyperledger Besu client. The RTFS also communicates with the Prometheus service to get the QoS measurements in real-time. This is accomplished by sending HTTP requests to the Prometheus REST API.

After successfully processing the QoE feedback, the verified QoE is fed to the Prometheus monitoring system using OpenTelemetry. Eventually, the MAESTRO service orchestrator retrieves the *'service QoE score'* metric from Prometheus using its REST API.

6. System prototype and evaluation

We have successfully deployed Feed4Cloud locally on our premises. The non-proprietary Feed4Cloud system components have been released as open-source and can be accessed at [72].

We hereafter briefly present the real-world deployment particulars, followed by a thorough evaluation of the RTFS layer.

6.1. Setup details

For demonstration purposes, we have set up a mock video streaming service hosted on a local server. In particular, we created a containerized application consisting of three modules: a Kafka producer, a Kafka consumer, and the rating widget. The Kafka producer captures frames from the given video file (i.e. *'countdown.avi'*). Meanwhile, the Kafka consumer runs a Flask application that reads data from the respective Kafka topic, provided via a URL parameter. The consumer decodes the JSON payload, enabling users to view the video content directly from their web browsers.

Service users can access the widget through their browsers at 'http://SERVICE_IP_ADDRESS/KAFKA_TOPIC'. To interact with the widget, users are required to sign in to their MetaMask wallet. That way, the submitted user ratings are automatically sent to the specified Hyperledger Besu node address of the Alastria Red B network where our smart contract has been deployed.

The rest of the components included in our prototype (excluding the smart contract) were deployed in UBITECH's premises. For this deployment we used 3 VMs, each one of which had 2 vCPUs, 4 GB RAM and 40 GB disk space. On the first VM, we deployed the video service components, accompanied by the rating widget and the Netdata agent. On the second VM, we deployed the Prometheus and MAESTRO services. The third VM hosted the RTFS service.

Finally, the RTFS credibility and reputation models – used for incorporating the QoS measurements in the scoring functions – were configured like so: In the expected QoE formula (1) the α constant is set to 4, whereas the β sensitivity parameter depends on the metric: for Network Throughput and Idle Jitter, it is 0.00025; for Packet Drops and Packet Errors, it is 0.025; for CPU usage, it is 0.01; and for Uptime, it is 10^{-9}. In the $Credibility_{user_{new}}$ formula, θ penalty and α adjustment factors are set to 0.5 and 1, respectively, while the weight w in $Credibility_{user}$ formula is 0.75. Lastly, the initial service reputation score is configured to be 3.5. To better motivate our choices for the initial credibility and reputation scores in the evaluation setting we investigated their impact indicatively for the Bayesian model in the full-on, sudden case.

Fig. 9 shows the actual rating submitted by the user in this case and the average expected QoE, while Fig. 10 depicts the inferred user credibility and service QoE score. For this evaluation, we experiment with different combinations of the lowest, median, and highest feasible values (i.e., 1, 3.5 and 5, respectively) for service reputation and user credibility, while adjusting the 7 automatically included ratings accordingly. Additionally, we assume the attack occurs 10 min after the newcomer joins the Feed4Cloud system.

As demonstrated in Fig. 10, the slander's credibility gets immediately minimized upon attack launching, irrespective of its prior value, as imposed by the penalty factor and the high w weight which results in the increased potency of the current behavior over the historical one. This measure ensures that logging from different accounts as a new user could not be exploited as a workaround for attackers as they get easily detected. In what concerns the impact on the service reputation, we show that the higher the initial credibility, the higher the value the reputation reaches before the attack, according to Eq. (5). Overall, since the user rating's impact on the service reputation is minimized during the attack, the only actual change is that the service QoE score is delayed in increasing. These observations corroborate that the cold start problem is indeed mitigated.

6.2. Real-world demonstration

The indicative screenshots shown in Fig. 11 give a glimpse of Feed4Cloud's real-world deployment. A video demonstration showcasing the Feed4Cloud functionalities is found at [73].

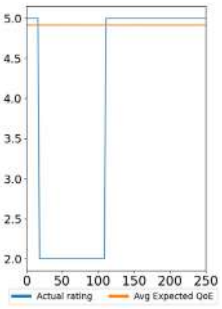
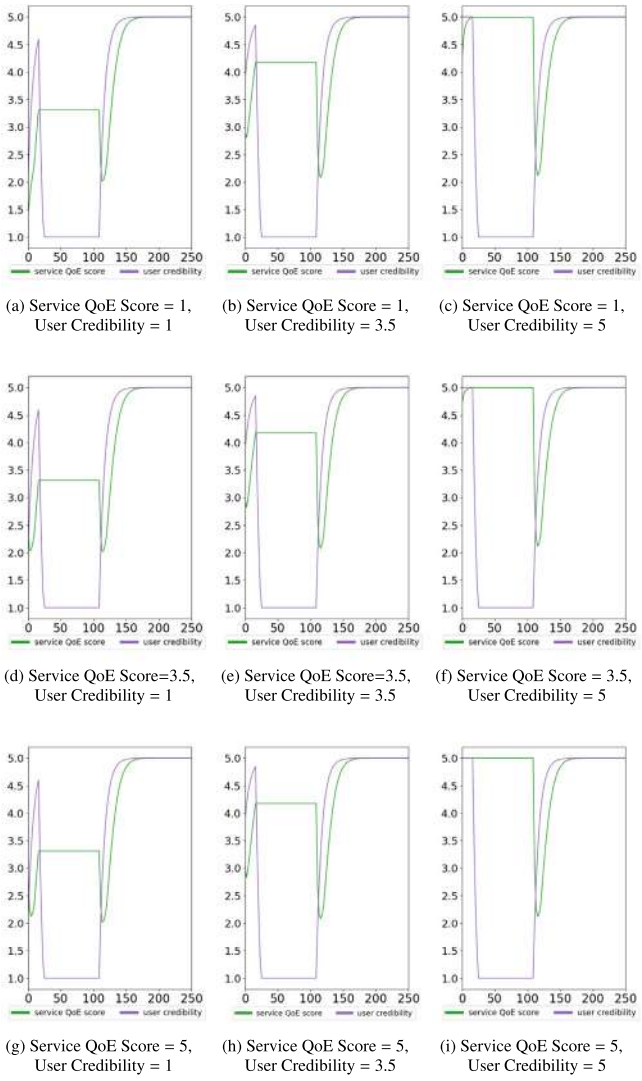


Fig. 9. Actual user rating & expected QoE in a full-on sudden attack.



(a) Service QoE Score = 1, User Credibility = 1
(b) Service QoE Score = 1, User Credibility = 3.5
(c) Service QoE Score = 1, User Credibility = 5
(d) Service QoE Score=3.5, User Credibility = 1
(e) Service QoE Score=3.5, User Credibility = 3.5
(f) Service QoE Score = 3.5, User Credibility = 5
(g) Service QoE Score = 5, User Credibility = 1
(h) Service QoE Score = 5, User Credibility = 3.5
(i) Service QoE Score = 5, User Credibility = 5

Fig. 10. Impact of different initial values.

Our web service features a designated online rating widget for service users to submit feedback on their perceived service's QoE performance (sub- Fig. 11(a)). This widget has been integrated with MetaMask to securely record the submitted ratings on the blockchain ledger in real-time (sub- Fig. 11(b)). Once the service evaluation has been completed, OpenTelemetry is used to collect the service reputation score, which is constantly monitored by Prometheus (sub- Fig. 11(c)). Based on the QoE feedback, service providers can adjust their service deployment, using SLO policies combining QoE and QoS values, and thus, ultimately QoE-aware service management is enabled.

6.3. RTFS evaluation

This section concerns the evaluation of RTFS, focusing on the behavior of its fundamental models, particularly the service reputation and the user credibility modules.

6.3.1. Experimental methodology

We designed a set of experiments to evaluate how the reputation and credibility modules react to different system conditions and attack tactics. In particular, we vary the following aspects:

- **Verification layer**: Our goal is to highlight the contribution of the verification layer. Thus, we evaluate how the service's reputation score is affected when relying solely on verified user ratings against lacking a feedback filtering mechanism.
- **Low feedback**: We consider two situations in which low feedback is provided: the first one involves a slander sending invalid feedback, while the second corresponds to an honest user reporting a genuine experience with degraded QoE.
- **Attack behavior**: We experiment with two different attack models, a full-on and a probabilistic one. On top of that, we consider two different ways of launching these attacks. In the first one, the attacker strikes abruptly, while in the second, we emulate a sneaky slander that exploits an ongoing system's QoS degradation to justify its negative feedback.

6.3.2. Experimental setup

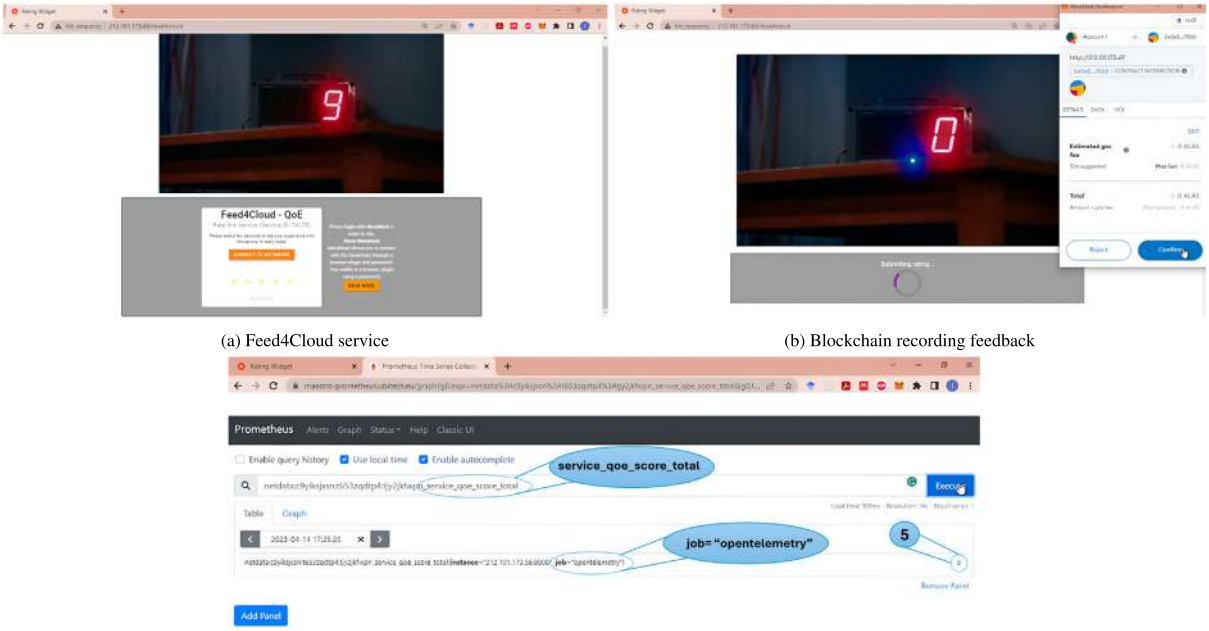
In our experiments, we utilized a JavaScript test script to automate the submission of user feedback. Our script simulates a user that sends a new rating to the blockchain-hosted smart contract every minute. The user was set to act honestly for a random period of time and then transition to an attacking mode for another random period. The attacking period was set to last approximately half the honest period.[4] Each experiment lasted for 10 h, collecting a total of 600 user ratings.

The test script sends ratings to the Alastria Red B node, where our smart contract is deployed, through web3.js. The sent ratings must originate from a specific blockchain wallet address. Thus, we assume that ratings are being sent by a single user, associated with the MetaMask wallet address that was used to log in on the client browser. That way, keeping track of the user credibility score fluctuations is facilitated, and the identification of the impact of the user behavior on the overall service reputation score is straightforward.

To assess the consistency of the RTFS's models under different system conditions and user behaviors, we conducted three rounds of experiments following the devised methodology 6.3.1:

- *Experiment 1*: Evaluating RTFS's effectiveness against sudden full-on and probabilistic attacks.
- *Experiment 2*: Evaluating RTFS in case of degraded QoS service performance. To simulate this scenario, we used the Apache JMeter tool [74] to fabricate an increased traffic load. Multiple service user access requests were manually generated at random intervals to overload the Feed4Cloud service host server and thus, artificially induce QoS impairments.
- *Experiment 3*: Combining the previous two experiments, this experiment aimed at evaluating RTFS against more sophisticated attacks. It involves sneaky full-on and probabilistic attacks that are launched during degraded service QoS performance to circumvent the reputation model.

[4] In fact, the attacking period ranged from 2.5 to 3.5 h, while the honest period ranged from 5.5 to 6.5 h.

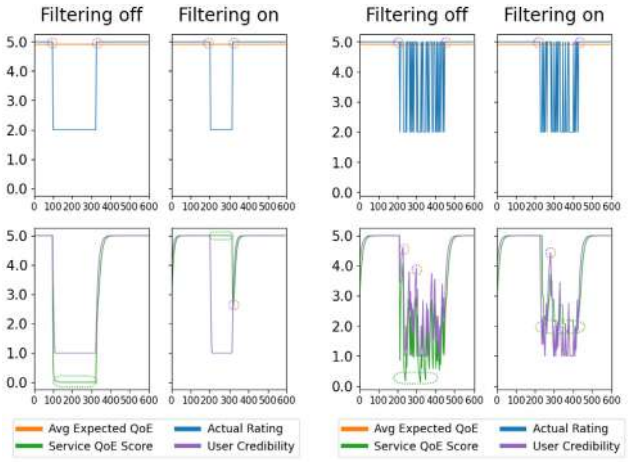


(a) Feed4Cloud service

(b) Blockchain recording feedback

(c) Prometheus monitoring service QoE

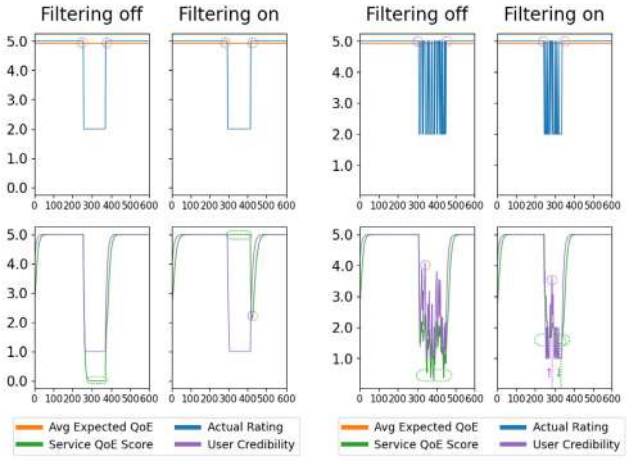
Fig. 11. Feed4Cloud system real-world demo.



(a) Full-on Attack

(b) Probabilistic Attack

Fig. 12. Experiment 1: Fuzzy Reputation Model Results in Sudden Attack.



(a) Full-on Attack

(b) Probabilistic Attack

Fig. 13. Experiment 1: Bayesian Reputation Model Results in Sudden Attack.

6.3.3. Results

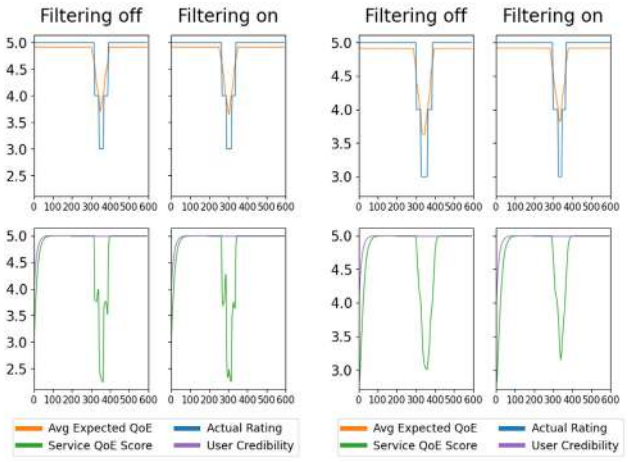
The evaluation results for both the *Fuzzy* and the *Bayesian* reputation model are presented in Figs. 12–16. Figs. 12 and 13 show the results of Experiment 1 for the two reputation models being used. Fig. 14 illustrates the results of Experiment 2. Lastly, Figs. 15 and 16 correspond to the results of the last Experiment. In all cases, the first-row sub-figures illustrate the '*Average Expected QoE*' (orange lines) and the '*Actual User Rating*' (blue lines), while the second-row sub-figures show the '*User Credibility*' and the '*Service QoE Score*' metrics, represented by purple and green lines, respectively.

It is very important to clarify that the '*Average Expected QoE*' is only used for visualization purposes to facilitate the results' comprehensibility. This metric is computed as the average value of the six expected QoE values inferred as described in Section 5.5.1. However, it is neither calculated nor used by the RTFS functional pipeline.

Our main objective was to demonstrate the benefits of the RTFS verification layer. To this end, we present the results for two distinct cases: with and without a filtering mechanism; hereafter, denoted as 'Filtering on' and 'Filtering off', respectively, and corresponding to a different column in the figures.

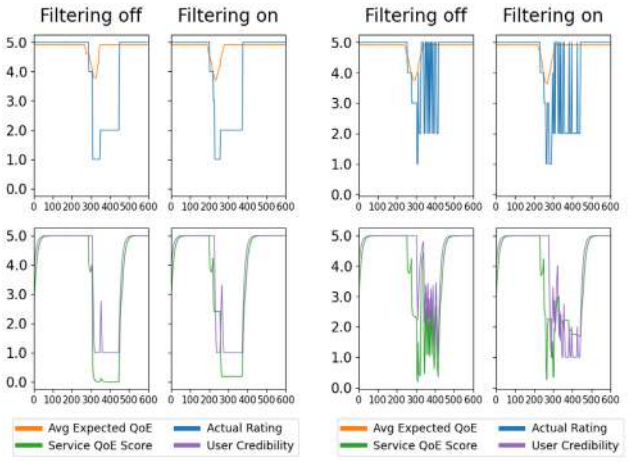
Experiment 1: We annotate Figs. 12 and 13 to show better the attack period (starting when the *'Actual Rating'* drops and ending when it recovers), the minimum *'User Credibility'* and *'Service QoE scores'* per case and any extraordinary temporary drops. The same trends apply in the following figures.

The results show that when every user rating is taken into account, regardless of its trustworthiness (i.e., 'Filtering off'), the service reputation score is more prone to fluctuations. However, with a proper filtering mechanism in place, it becomes harder for any deliberate attempts to undermine the service's reputation, to succeed. This is



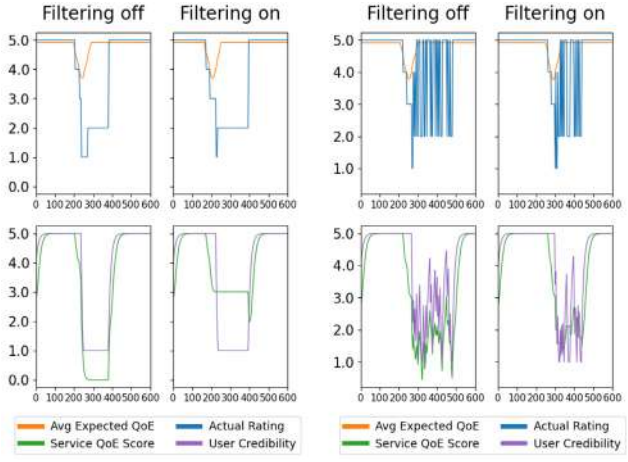
(a) Fuzzy Reputation Model (b) Bayesian Reputation Model

Fig. 14. Experiment 2: Degraded service QoE.



(a) Full-on Attack (b) Probabilistic Attack

Fig. 15. Experiment 3: Fuzzy Reputation Model Results in Sneaky Attack.



(a) Full-on Attack (b) Probabilistic Attack

Fig. 16. Experiment 3: Bayesian Reputation Model Results in Sneaky Attack.

especially true in cases of full-on attacks, wherein the attacker's misbehaving is easily identifiable. For example, for the fuzzy reputation model without a filtering phase, when subjected to a sudden full-on attack (launched at around 100 s and ending at 300 s as depicted in circles in 'Filtering off' in 12(a)), the service reputation score was significantly decreased (green oval). In contrast, when the same attack occurs, but the filtering mechanism intervenes (taking place from a bit after 200 s to a bit after 300 s in 'Filtering on' of 12(a)), the service reputation score is not affected at all (green oval). In this case, there is only a temporary drop (bicolor circle) in the service QoE score (a bit after 300 s, at the end of the attacking period). That is because the user begins to behave honestly, sending high ratings close to the expected QoE, which results in increasing its credibility and thus, showing a higher impact of both its current and past ratings on the final service score (according to Eq. (5)). The Bayesian model responds to the attack in a similar fashion, with the respective service score being almost zero when there is no filtering (green oval in 'Filtering off' sub-figures of 13(a)), but remaining unaffected when fraudulent feedback has been discarded (green oval 'Filtering on' sub-figures of 13(a)). These results demonstrate that fairness in service evaluation is completely secured against sudden full-on attacks by the RTFS filtering functionality as opposed to when the latter is absent.

It should be noted that a probabilistic attack might still manipulate the service's reputation, to some degree, as the attacker mixes legitimate with roguish behavior. As a consequence, the user credibility is increased at times and gains a higher impact on the calculation of the service QoE score (see Eq. (5)). In both the fuzzy and the Bayesian models, the service QoE score reaches lower values in cases 12(b) and 13(b) as compared to the respective full-on attack cases, wherein it is not affected at all (as seen when comparing the green ovals). For instance, in 13(b), service reputation drops even at around 1.5 in $t = 300$ s. That is because the user credibility was quite high a few moments ago (being almost 4 in-between 200 s – 300 s), thus allowing for influencing the service reputation as $Reputation_{service_{new}}$ prevails in Eq. (5) (depicted with purple and green circles and corresponding arrows). Still, in the case of Bayesian, this metric incorporates previous ratings as well(Eq. (4)). On the contrary, the fuzzy calculation relies completely on the latest ratings(Eq. (6)), thus maintaining higher service scores, as illustrated in sub- Fig. 12(b). However, compared to 'Filtering off' figures, the results prove the value added by the verification layer.

Experiment 2: As depicted in Fig. 14, the QoS degradation is happening at around 300 s when the expected QoE drops, and lasts about 100 s for both reputation models. Based on the sub-figures of the first row, we observe that the actual user rating is aligned with the average expected QoE as the blue line follows the inclination of the orange one. Both models successfully recognize that the negative feedback stems from the actual degraded QoE experience. As a consequence, the calculated service QoE score is fairly lowered, whereas the user's credibility remains unchanged (depicted by the green and purple lines, respectively). Therefore, both the credibility and the reputation models behave in a rational manner.

Experiment 3: A furtive attack could outwit the reputation system for a while before being detected. Zooming into the 'Filtering on' case, there is still improvement when compared to the 'Filtering off' case. Indeed, the results revealed interesting information about the effectiveness of the two employed reputation models when tested against sneaky attackers. While both models can detect and exclude false ratings in straightforward attacks, the fuzzy model is more vulnerable to sneaky attackers. This was demonstrated through a temporary system QoS degradation, at around 450 s; that caused an increase in the attacker's credibility as its negative ratings passed the verification, which in turn increased its impact on service evaluation and allowed the attacker to harm the service's reputation (Fig. 15(a)). On the other hand, the Bayesian model managed to defend against this occurrence (Fig. 16(a), at t≈250s). The reason behind this is the difference of the two models

when inferring $Reputation_{service_{new}}$. Unlike the fuzzy approach which assesses the current behavior only based on the latest rating, the Bayesian reputation formula (Eq. (4)) considers the historical reputation information, thus limiting the impact of the latest ratings. As a result, even though this time the service QoE score is lower, this is because of the preceding QoS degradation, when the low ratings are trustworthy and hence verified and counted in. Therefore, the contribution of the verification layer lies in the stability of the reputation score when invalid ratings come into play, rather than the value of this score per se.

Even in the case of sneaky probabilistic attacks (Figs. 15(b) and 16(b)), the inclusion of a verification layer can still alleviate unjustified low reputation scores, at least to some extent. However, among the considered attack tactics, this is the hardest to detect and deal with; advanced detection mechanisms might be better suited to accurately identify sneaky probabilistic attacks. We discuss how this aspect can be addressed in Section 7.2.2.

Overall, the results confirmed the usefulness of the verification layer. RTFS quickly identifies false negative feedback and filters it off from the service evaluation process. As a result, it effectively combats any malevolent attempts to defame the service. Moreover, the evaluation validated that the employed reputation and credibility models exhibit a desired behavior since they treat users and evaluate services fairly. Dishonest users are penalized and trustworthy user feedback is prioritized.

6.3.4. Identified limitations

Although the aforementioned evaluation allows for achieving our objectives to demonstrate the benefits of introducing the verification intermediate layer and determine whether the reputation and credibility models behave as expected, we acknowledge that it entails the following limitations:

- Due to the requirement of alignment of the user rating with most expected QoE values, (see Section 5.5.2), the verification process might exclude honest low ratings when for example, one or two, QoS metrics are impaired. Still, we have found that when a real QoS degradation occurs, the expected QoE accurately reflects it. Thus, it is less likely that only a few QoS measurements are affected.
- Our current models underperform against sneaky probabilistic attacks, highlighting the need for more advanced detection approaches to deal with such sophisticated attack strategies.
- When the attacker reverts to being honest its prior negative ratings temporarily lower the reputation score. To mitigate this effect, fine-tuning the weights in Eq. (5) is necessary.
- If malicious users outnumber honest ones and launch probabilistic attacks simultaneously, the reputation model may fail to be as resilient. Still, this is unlikely to happen in a real rating system. Moreover, requiring users to sign in using a Metamask wallet address already makes it harder for attackers to create many fake identities to launch such an attack.

7. Discussion

7.1. Resilience to attacks on the reputation system

Provided that reputation systems can be plagued by several security issues, many security considerations might arise concerning Feed4Cloud's attack resilience. Considering that Feed4Cloud is strongly dependent on user-provided feedback, we focused on providing a system resilient primarily against 'slandering' attacks [75], in which malicious users might collude to provide fake negative feedback. However, we argue that the proposed Feed4Cloud framework, thanks to its native features, is inherently resilient against other well-known types of attacks on reputation systems, such as Sybil and Man in the Middle (MITM) attacks as well. Feed4Cloud's ability to cope with the aforementioned misbehaviors is attributed mainly to the following aspects:

- User authentication mechanisms in place: Feed4Cloud requires users to be registered and authenticated (via signing in to their Metamask wallet) before they are permitted to submit ratings. Such a requirement makes it very difficult for Sybil attackers to create multiple fake identities and overall, allows for restricting access for unauthorized and potentially malicious users, thus providing additional security assurances.
- Feedback verification and high penalty in user credibility formula: Besides filtering out any fake ratings, the high penalty imposed by the 'linear increase, exponential decrease' principle decreases significantly user credibility in case of fake feedback detection, let alone in case of multiple fake feedback submissions. Hence, not only fake feedback of either distinctly malicious users or even camouflaged ones (e.g. Sybils or MITM attackers) will be excluded, but also their credibility scores will be quickly minimized. Even attackers using sophisticated tactics, such as on–off malicious actors who mix honest with fake feedback to slip undetected through the reputation system, will fail as they will be strongly punished when misbehaving while recovering will be arduous.
- Credibility-aware and weighted service reputation score evaluation: Since the reputation model considers the user credibility but also the past reputation scores in the reputation formula, even if particular fake ratings are falsely accepted, their impact on the service evaluation will be minor if they are generated from a user with low credibility. Therefore, eventually, inconsiderable or no harm at all, will be caused.
- Blockchain-ensured feedback integrity: Feed4Cloud provides an extra layer of security by using Blockchain for recording user feedback. That way, it becomes impossible to modify the feedback to manipulate the evaluation result.

7.2. Future directions

7.2.1. Applicability areas

Feed4Cloud provides a solution mainly focused on supporting QoE-aware cloud service management and orchestration, which goes beyond today's QoS-based approaches [76]. By taking into account user experience, our approach aims to enable sophisticated approaches for resource allocation [77]. For instance, a DevOps user can design elasticity policies and actions that incorporate QoE feedback in the form of custom metrics.

Moreover, such an approach can be particularly important in multi-cloud, geo-distributed setups by allowing service providers to collect fine-grained feedback about different deployed instances of the same service. Since those instances can be located in different regions and present heterogeneous performance characteristics, the experience may vary significantly between different users, based on their location and providers. Our approach can quantify this difference in experience and allows its use by the orchestration system to enhance service delivery.

In a similar vein, Feed4Cloud can be leveraged to assist in decentralized SLA management. More specifically, QoE performance targets could be defined (e.g. using the OpenSLO specification) and then, the trustworthy service QoE scores exposed by Feed4Cloud can be used to monitor SLO compliance and trigger QoE-based alert policies. This can be further consolidated by relying on blockchain. This is feasible via specialized smart contracts, which will, for example, automatically increase the service's reputation score when the measured QoS metrics adhere to the predetermined SLA conditions or impose penalties in case of SLO violations. That way, Feed4Cloud can contribute towards facilitating trust establishment between mutually untrusted contracting parties in cloud settings.

We argue that by integrating user feedback into real-world cloud services, Feed4Cloud opens up the potential for other use cases, as well. One case could be the collection of user-generated feedback on different versions of the same service. This feedback can be used by service providers to assess the reception of new service features by users and significantly improve software engineering operations such as A/B testing [78], canary releases, progressive rollouts, and overall CI/CD pipelines [79]. We plan to further explore this in the future.

7.2.2. Potential extensions

Building upon its fundamental blocks, Feed4Cloud can be further extended to provide even more advanced functionalities. For instance, regarding feedback verification, we are currently using a one-on-one approach, i.e. comparing a particular rating with the corresponding QoS at a specific time point. One potential enhancement is the usage of machine learning-based change point detection (CPD) models that would inspect the QoS and user QoE time series in a wider time frame, to detect asynchronous changes and thus, identify invalid QoE input [25,80]. This can be particularly useful in cases of complex attack tactics, such as sneaky probabilistic attacks.

Another issue concerns user identities. In our current implementation, we rely on the default digital wallet addresses and do not include any further personally identifiable information. Such an approach hides the identity of the individual unless they are linked to additional identifiers and can therefore be generally considered pseudonymous. That way, we ensure user privacy. To reinforce this measure, we envision incorporating advanced identity verification mechanisms, such as Self-sovereign identities (SSI), and we are specifically considering ALASTRIA Decentralized Identifier Model [81]. Such an extension will allow for managing credentials in a decentralized and secure way since they are stored on a distributed ledger and only accessed when required. In addition, this will provide an extra layer of security as it will make it even more difficult to create fake identities or to compromise the credentials of legitimate ones. As of the time of writing, although SSI constitutes an emerging technology, its effective integration with user feedback collection is still an open issue to be explored [82].

Last but not least, since trustworthy user feedback is vital for Feed4Cloud, another interesting direction is to introduce an incentive mechanism to encourage the participation of users in the evaluation with honest feedback. In this regard, the Feed4Cloud credibility scheme can be extended to support a reputation-based incentive mechanism [83–85]. More specifically, incentive-based user behavior control can be achieved by associating the user credibility values with certain privileges. For example, potential incentives that can be given to motivate trustworthy behavior include gaining revenues for offering trustworthy QoE feedback [42,86], accessing more services as credibility increases [37] and/or prioritizing trustworthy users during resource orchestration to guarantee meeting the corresponding SLA requirements [87].

8. Conclusions

In this paper, we introduced Feed4Cloud: a modular system that considers user QoE as an integral feature of the cloud monitoring plane, while allowing only for trustworthy user feedback to be considered in cloud-to-edge orchestration. Our solution encompasses well-established QoS-to-QoE correlation models to support QoE verification effectively, as well as appropriate service evaluation models to assist QoE-based service management. The presented evaluation results substantiated the added value of the introduced verification layer. Feed4Cloud ensures user feedback is collected and stored in a transparent and secure way through the use of Blockchain. For our prototype implementation, we provided a custom oracle smart contract deployed on Alastria Red B network. In addition, we interconnected our system with MAESTRO, a real-world orchestrator to enable feedback-aware service monitoring and configuration.

CRediT authorship contribution statement

Ioanna Angeliki Kapetanidou: Writing – review & editing, Writing – original draft, Visualization, Software, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization. **Christos-Alexandros Sarros:** Writing – review & editing, Writing – original draft, Visualization, Methodology, Funding acquisition, Formal analysis, Conceptualization. **Giannis Ledakis:** Writing – review & editing, Methodology, Funding acquisition, Conceptualization. **Vassilis Tsaoussidis:** Writing – review & editing, Project administration, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The link to Feed4Cloud Gitlab repository has been shared.

Acknowledgments

This work was supported by the European Union's Horizon 2020 research and innovation programme [grant number 957228]; and the European Regional Development Fund of the European Union and Greek National funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call RESEARCH – CREATE – INNOVATE [project code:T2EDK-02039].

References

[1] Y. Verginadis, A review of monitoring probes for cloud computing continuum, in: L. Barolli (Ed.), Advanced Information Networking and Applications, Springer International Publishing, Cham, 2023, pp. 631–643.

[2] I. Rec, P. 10: Vocabulary for performance and quality of service, amendment 2: New definitions for inclusion in recommendation ITU-T P. 10/G. 100, Int. Telecommun. Union Geneva (2008) 10.

[3] J. Arellano-Uson, E. Magaña, D. Morato, M. Izal, Survey on quality of experience evaluation for cloud-based interactive applications, Appl. Sci. 14 (5) (2024) 1987.

[4] M. Gramaglia, I. Digon, V. Friderikos, D. von Hugo, C. Mannweiler, M.A. Puente, K. Samdanis, B. Sayadi, Flexible connectivity and QoE/QoS management for 5G networks: The 5G NORMA view, in: 2016 IEEE International Conference on Communications Workshops, ICC, 2016, pp. 373–379, http://dx.doi.org/10.1109/ICCW.2016.7503816.

[5] T. Wang, P. Wang, S. Cai, Y. Ma, A. Liu, M. Xie, A unified trustworthy environment establishment based on edge computing in industrial IoT, IEEE Trans. Ind. Inform. 16 (9) (2020) 6083–6091, http://dx.doi.org/10.1109/TII.2019.2955152.

[6] H. Hassan, A.I. El-Desouky, A. Ibrahim, E.-S.M. El-Kenawy, R. Arnous, Enhanced QoS-based model for trust assessment in cloud computing environment, IEEE Access 8 (2020) 43752–43763, http://dx.doi.org/10.1109/ACCESS.2020.2978452.

[7] A.A. Laghari, X. Zhang, Z.A. Shaikh, A. Khan, V.V. Estrela, S. Izadi, A review on quality of experience (QoE) in cloud computing, J. Reliab. Intell. Environ. 10 (2) (2024) 107–121.

[8] C.E. Rothenberg, D.A. Lachos Perez, N.F. Saraiva de Sousa, R.V. Rosa, R.U. Mustafa, M.T. Islam, P.H. Gomes, Intent-based control loop for DASH video service assurance using ML-based edge QoE estimation, in: 2020 6th IEEE Conference on Network Softwarization, NetSoft, 2020, pp. 353–355, http://dx.doi.org/10.1109/NetSoft48620.2020.9165375.

[9] A.A. Laghari, H. He, A. Khan, N. Kumar, R. Kharel, Quality of experience framework for cloud computing (QoC), IEEE Access 6 (2018) 64876–64890, http://dx.doi.org/10.1109/ACCESS.2018.2865967.

[10] A. Sackl, R. Schatz, T. Hossfeld, F. Metzger, D. Lister, R. Irmer, Qoe management made uneasy: The case of cloud gaming, in: 2016 IEEE International Conference on Communications Workshops, ICC, 2016, pp. 492–497, http://dx.doi.org/10.1109/ICCW.2016.7503835.

[11] B. Wang, M. Li, X. Jin, C. Guo, A reliable IoT edge computing trust management mechanism for smart cities, IEEE Access 8 (2020) 46373–46399, http://dx.doi.org/10.1109/ACCESS.2020.2979022.

[12] S. Dutta, T. Taleb, A. Ksentini, QoE-aware elasticity support in cloud-native 5G systems, in: 2016 IEEE International Conference on Communications, ICC, 2016, pp. 1–6, http://dx.doi.org/10.1109/ICC.2016.7511377.

[13] M. Aazam, K.A. Harras, S. Zeadally, Fog computing for 5G tactile industrial internet of things: QoE-aware resource allocation model, IEEE Trans. Ind. Inform. 15 (5) (2019) 3085–3092, http://dx.doi.org/10.1109/TII.2019.2902574.

[14] Qualinet WG1 task force on "managing web and cloud QoE", 2023, https://www.qualinet.eu/task-forces/. (Accessed 29 March 2023).

[15] D.N. da Hora, A.S. Asrese, V. Christophides, R. Teixeira, D. Rossi, Narrowing the gap between QoS metrics and web QoE using above-the-fold metrics, in: Passive and Active Measurement: 19th International Conference, PAM 2018, Berlin, Germany, March 26–27, 2018, Proceedings 19, Springer, 2018, pp. 31–43.

[16] T. Hoßfeld, P.E. Heegaard, L. Skorin-Kapov, M. Varela, Fundamental relationships for deriving QoE in systems, in: 2019 Eleventh International Conference on Quality of Multimedia Experience, QoMEX, 2019, pp. 1–6, http://dx.doi.org/10.1109/QoMEX.2019.8743339.

[17] T. Hoßfeld, P. Tran-Gia, M. Fiedler, Quantification of quality of experience for edge-based applications, in: Managing Traffic Performance in Converged Networks: 20th International Teletraffic Congress, ITC20 2007, Ottawa, Canada, June 17-21, 2007. Proceedings, Springer, 2007, pp. 361–373.

[18] M. Fiedler, T. Hossfeld, P. Tran-Gia, A generic quantitative relationship between quality of experience and quality of service, IEEE Netw. 24 (2) (2010) 36–41, http://dx.doi.org/10.1109/MNET.2010.5430142.

[19] I. Estimating end-to-end, Performance in IP networks for data applications, 2014.

[20] M. Tang, X. Dai, J. Liu, J. Chen, Towards a trust evaluation middleware for cloud service selection, Future Gener. Comput. Syst. 74 (2017) 302–312.

[21] K. Papadakis-Vlachopapadopoulos, R.S. González, I. Dimolitsas, D. Dechouniotis, A.J. Ferrer, S. Papavassiliou, Collaborative SLA and reputation-based trust management in cloud federations, Future Gener. Comput. Syst. 100 (2019) 498–512, http://dx.doi.org/10.1016/j.future.2019.05.030, URL https://www.sciencedirect.com/science/article/pii/S0167739X18329248.

[22] L. Qu, Y. Wang, M.A. Orgun, L. Liu, H. Liu, A. Bouguettaya, CCCloud: Context-aware and credible cloud service selection based on subjective assessment and objective assessment, IEEE Trans. Serv. Comput. 8 (3) (2015) 369–383, http://dx.doi.org/10.1109/TSC.2015.2413111.

[23] D. Marudhadevi, V.N. Dhatchayani, V.S. Sriram, A trust evaluation model for cloud computing using service level agreement, Comput. J. 58 (10) (2015) 2225–2232.

[24] K. Hamadache, S. Rizou, Holistic SLA ontology for cloud service evaluation, in: 2013 International Conference on Advanced Cloud and Big Data, 2013, pp. 32–39, http://dx.doi.org/10.1109/CBD.2013.18.

[25] S. Wang, Z. Zheng, Z. Wu, M.R. Lyu, F. Yang, Reputation measurement and malicious feedback rating prevention in web service recommendation systems, IEEE Trans. Serv. Comput. 8 (5) (2015) 755–767, http://dx.doi.org/10.1109/TSC.2014.2320262.

[26] S. Siadat, A.M. Rahmani, H. Navid, Identifying fake feedback in cloud trust management systems using feedback evaluation component and Bayesian game model, J. Supercomput. 73 (2017) 2682–2704.

[27] V. Viji Rajendran, S. Swamynathan, Hybrid model for dynamic evaluation of trust in cloud services, Wirel. Netw. 22 (2016) 1807–1818.

[28] T.H. Noor, Q.Z. Sheng, L. Yao, S. Dustdar, A.H. Ngu, CloudArmor: Supporting reputation-based trust management for cloud services, IEEE Trans. Parallel Distrib. Syst. 27 (2) (2016) 367–380, http://dx.doi.org/10.1109/TPDS.2015.2408613.

[29] R. Nagarajan, R. Thirunavukarasu, S. Shanmugam, A fuzzy-based intelligent cloud broker with MapReduce framework to evaluate the trust level of cloud services using customer feedback, Int. J. Fuzzy Syst. 20 (2018) 339–347.

[30] F.G. Mármol, M.Q. Kuhnen, Reputation-based web service orchestration in cloud computing: A survey, Concurr. Comput.: Pract. Exper. 27 (9) (2015) 2390–2412, http://dx.doi.org/10.1002/cpe.3177, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.3177. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.3177.

[31] M. Si, Q. Li, Shilling attacks against collaborative recommender systems: a review, Artif. Intell. Rev. 53 (2020) 291–319.

[32] W. Zhou, J. Wen, Q. Qu, J. Zeng, T. Cheng, Shilling attack detection for recommender systems based on credibility of group users and rating time series, PLoS One 13 (5) (2018) e0196533.

[33] L. Qu, Y. Wang, M.A. Orgun, Cloud service selection based on the aggregation of user feedback and quantitative performance assessment, in: 2013 IEEE International Conference on Services Computing, 2013, pp. 152–159, http://dx.doi.org/10.1109/SCC.2013.92.

[34] T. Hobfeld, R. Schatz, M. Varela, C. Timmerer, Challenges of QoE management for cloud applications, IEEE Commun. Mag. 50 (4) (2012) 28–36, http://dx.doi.org/10.1109/MCOM.2012.6178831.

[35] D. Chemodanov, P. Calyam, S. Valluripally, H. Trinh, J. Patman, K. Palaniappan, On QoE-oriented cloud service orchestration for application providers, IEEE Trans. Serv. Comput. 14 (4) (2021) 1194–1208, http://dx.doi.org/10.1109/TSC.2018.2866851.

[36] J.M.J. Valero, V. Theodorou, M.G. Pérez, G.M. Pérez, SLA-driven trust and reputation management framework for 5G distributed service marketplaces, IEEE Trans. Dependable Secure Comput. (2023) 1–13, http://dx.doi.org/10.1109/TDSC.2023.3292589.

[37] R. Fernando, R. Ranchal, B. Bhargava, P. Angin, A monitoring approach for policy enforcement in cloud services, in: 2017 IEEE 10th International Conference on Cloud Computing, CLOUD, 2017, pp. 600–607, http://dx.doi.org/10.1109/CLOUD.2017.82.

[38] Y. Alowayed, M. Canini, P. Marcos, M. Chiesa, M. Barcellos, Picking a partner: A fair blockchain based scoring protocol for autonomous systems, in: Proceedings of the Applied Networking Research Workshop, 2018, pp. 33–39.

[39] R. Ranchal, O. Choudhury, SLAM: A framework for SLA management in multi-cloud ecosystem using blockchain, in: 2020 IEEE Cloud Summit, IEEE, 2020, pp. 33–38.

[40] K.M. Khan, J. Arshad, W. Iqbal, S. Abdullah, H. Zaib, Blockchain-enabled real-time SLA monitoring for cloud-hosted services, Cluster Comput. (2022) 1–23.

[41] H. Zhou, X. Ouyang, Z. Ren, J. Su, C. de Laat, Z. Zhao, A blockchain based witness model for trustworthy cloud service level agreement enforcement, in: IEEE INFOCOM 2019-IEEE Conference on Computer Communications, IEEE, 2019, pp. 1567–1575.

[42] R.B. Uriarte, H. Zhou, K. Kritikos, Z. Shi, Z. Zhao, R. De Nicola, Distributed service-level agreement management with smart contracts and blockchain, Concurr. Comput.: Pract. Exper. 33 (14) (2021) e5800.

[43] T.H. Noor, Q.Z. Sheng, A. Alfazi, Reputation attacks detection for effective trust assessment among cloud services, in: 2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, IEEE, 2013, pp. 469–476.

[44] Chainlink - oracle definition, 2024, https://chain.link/education/blockchain-oracles#:~:text=DEFINITION,legacy%20systems%2C%20and%20advanced%20computations. (Accessed 18 September 2024).

[45] M.M. Shabir, S.M. Danish, K. Zhang, BlockQoS: Fair monetization of on-demand quality-of-service using blockchains, Distrib. Ledger Technol.: Res. Pract. 2 (2) (2023) 1–25.

[46] K. Meng, Z. Wu, M. Bilal, X. Xia, X. Xu, Blockchain-enabled decentralized service selection for QoS-aware cloud manufacturing, Expert Syst. (2024) e13602.

[47] N. Neeraj, A. Nellikeri, P. Varun, S. Reddy, M. Shanbhag, D. Narayan, A. Husain, Service level agreement violation detection in multi-cloud environment using ethereum blockchain, in: 2023 International Conference on Networking and Communications, ICNWC, IEEE, 2023, pp. 1–7.

[48] H. Al-Breiki, M.H.U. Rehman, K. Salah, D. Svetinovic, Trustworthy blockchain oracles: Review, comparison, and open research challenges, IEEE Access 8 (2020) 85675–85685, http://dx.doi.org/10.1109/ACCESS.2020.2992698.

[49] A. Battah, Y. Iraqi, E. Damiani, Blockchain-based reputation systems: Implementation challenges and mitigation, Electronics 10 (3) (2021) http://dx.doi.org/10.3390/electronics10030289, URL https://www.mdpi.com/2079-9292/10/3/289.

[50] Vue.js - the progressive JavaScript framework | vue.js, 2023, https://vuejs.org/. (Accessed 3 April 2023).

[51] MetaMask, 2023, https://metamask.io/. (Accessed 3 April 2023).

[52] Web3.js library, 2023, https://github.com/web3/web3.js. (Accessed 3 April 2023).

[53] Netdata, 2023, https://github.com/netdata/netdata. (Accessed 3 April 2023).

[54] Prometheus.io, 2023, https://prometheus.io/. (Accessed 29 March 2023).

[55] OpenTelemetry, 2023, https://opentelemetry.io/. (Accessed 3 April 2023).

[56] MAESTRO orchestrator, 2023, https://themaestro.ubitech.eu/. (Accessed 3 April 2023).

[57] Soliditi programming language, 2023, https://soliditylang.org/. (Accessed 3 April 2023).

[58] Ganache, 2023, https://trufflesuite.com/docs/ganache/. (Accessed 29 March 2023).

[59] Truffle, 2023, https://trufflesuite.com/. (Accessed 29 March 2023).

[60] ALASTRIA red B network, 2023, https://github.com/alastria/alastria-node-besu. (Accessed 29 March 2023).

[61] T. Hoßfeld, P.E. Heegaard, M. Varela, L. Skorin-Kapov, M. Fiedler, From QoS distributions to QoE distributions: A system's perspective, in: 2020 6th IEEE Conference on Network Softwarization, NetSoft, IEEE, 2020, pp. 51–56.

[62] V. Jacobson, Congestion avoidance and control, ACM SIGCOMM Comput. Commun. Rev. 18 (4) (1988) 314–329.

[63] D. Wu, Z. Xu, B. Chen, Y. Zhang, What if routers are malicious? Mitigating content poisoning attack in NDN, in: 2016 IEEE Trustcom/BigDataSE/ISPA, 2016, pp. 481–488, http://dx.doi.org/10.1109/TrustCom.2016.0100.

[64] I.A. Kapetanidou, M. Hassan, C.-A. Sarros, M. Conti, V. Tsaoussidis, Reputation-based trust: A robust mechanism for dynamic adaptive streaming over named data networking, in: 2020 IEEE 14th International Conference on Big Data Science and Engineering, BigDataSE, 2020, pp. 114–121, http://dx.doi.org/10.1109/BigDataSE50710.2020.00023.

[65] Trustpilot, 2023, https://www.trustpilot.com/trust. (Accessed 28 September 2023).

[66] S. Kubler, J. Robert, W. Derigent, A. Voisin, Y. Le Traon, A state-of the-art survey & testbed of fuzzy AHP (FAHP) applications, Expert Syst. Appl. 65 (2016) 398–422.

[67] S. Wang, L. Sun, Q. Sun, J. Wei, F. Yang, Reputation measurement of cloud services based on unstable feedback ratings, Int. J. Web Grid Serv. 11 (4) (2015) 362–376.
[68] J. Mitchell, S. Rizvi, J. Ryoo, A fuzzy-logic approach for evaluating a cloud service provider, in: 2015 1st International Conference on Software Security and Assurance, ICSSA, 2015, pp. 19–24, http://dx.doi.org/10.1109/ICSSA.2015.014.
[69] Ritu, S. Jain, A trust model in cloud computing based on fuzzy logic, in: 2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology, RTEICT, 2016, pp. 47–52, http://dx.doi.org/10.1109/RTEICT.2016.7807780.
[70] A. Selvaraj, S. Sundararajan, Evidence-based trust evaluation system for cloud services using fuzzy logic, Int. J. Fuzzy Syst. 19 (2017) 329–337.
[71] Web3.py library, 2023, https://github.com/ethereum/web3.py. (Accessed 29 March 2023).
[72] Feed4Cloud gitlab repo, 2023, https://gitlab.com/groups/feed4cloud-trublo. (Accessed 30 March 2023).
[73] Feed4Cloud video demonstration, 2023, https://drive.google.com/drive/folders/1l1hbxd8QIO9KbonNYE03ix0jT6DIK263?usp=share_link. (Accessed 3 April 2023).
[74] Apache JMeter, 2023, https://jmeter.apache.org/. (Accessed 10 October 2023).
[75] K. Hoffman, D. Zage, C. Nita-Rotaru, A survey of attacks on reputation systems, 2007.
[76] A. Tsagkaropoulos, Y. Verginadis, N. Papageorgiou, F. Paraskevopoulos, D. Apostolou, G. Mentzas, Severity: a QoS-aware approach to cloud application elasticity, J. Cloud Comput. 10 (1) (2021) 45, http://dx.doi.org/10.1186/s13677-021-00255-5.
[77] S. Nádas, L. Ernström, L. Szilágyi, G. Patra, D. Krylov, J. Lynam, To QoE or not to QoE, in: Proceedings of the 2024 Applied Networking Research Workshop, 2024, pp. 38–44.
[78] R. Ros, P. Runeson, Continuous experimentation and a/b testing: A mapping study, in: Proceedings of the 4th International Workshop on Rapid Continuous Software Engineering, RCoSE '18, Association for Computing Machinery, New York, NY, USA, 2018, pp. 35–41, http://dx.doi.org/10.1145/3194760.3194766.
[79] C. Singh, N.S. Gaba, M. Kaur, B. Kaur, Comparison of different CI/CD tools integrated with cloud platform, in: 2019 9th International Conference on Cloud Computing, Data Science & Engineering, Confluence, 2019, pp. 7–12, http://dx.doi.org/10.1109/CONFLUENCE.2019.8776985.
[80] S. Nilizadeh, H. Aghakhani, E. Gustafson, C. Kruegel, G. Vigna, Think outside the dataset: Finding fraudulent reviews using cross-dataset analysis, in: The World Wide Web Conference, WWW '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 3108–3115, http://dx.doi.org/10.1145/3308558.3313647.
[81] Self-sovereign identity (SSI) implementation on alastria network, 2023, https://github.com/alastria/alastria-identity. (Accessed 30 March 2023).
[82] R. Soltani, U.T. Nguyen, A. An, A survey of self-sovereign identity ecosystem, Secur. Commun. Netw. 2021 (2021) 1–26.
[83] R. Han, Z. Yan, X. Liang, L.T. Yang, How can incentive mechanisms and blockchain benefit with each other? a survey, ACM Comput. Surv. 55 (7) (2022) 1–38.
[84] C.-A. Sarros, I.A. Kapetanidou, V. Tsaoussidis, Incentivising honest behaviour in P2P networks using blockchain-based reputation, in: 2021 Eighth International Conference on Software Defined Systems, SDS, 2021, pp. 1–6, http://dx.doi.org/10.1109/SDS54264.2021.9732162.
[85] T. Papaioannou, G. Stamoulis, An incentives' mechanism promoting truthful feedback in peer-to-peer systems, in: CCGrid 2005. IEEE International Symposium on Cluster Computing and the Grid, 2005, Vol. 1, 2005, pp. 275–283, http://dx.doi.org/10.1109/CCGRID.2005.1558565, Vol. 1.
[86] H. Zhou, X. Ouyang, Z. Ren, J. Su, C. de Laat, Z. Zhao, A blockchain based witness model for trustworthy cloud service level agreement enforcement, in: IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, 2019, pp. 1567–1575, http://dx.doi.org/10.1109/INFOCOM.2019.8737580.
[87] M. Macías, J. Guitart, Analysis of a trust model for SLA negotiation and enforcement in cloud markets, Future Gener. Comput. Syst. 55 (2016) 460–472.

Ioanna-Angeliki Kapetanidou (female) is a Ph.D. candidate in the Department of Electrical and Computer Engineering Department of Democritus University of Thrace (DUTH) under the supervision of Prof. V. Tsaoussidis and also a research associate in Athena Research and Innovation Center. She has been involved in several EU-funded research projects and co-authored many published papers and technical deliverables. Her research interests lie mainly in the areas of trust management and future generation networking technologies.

Christos-Alexandros Sarros (male) is a Distributed Systems Researcher in UBITECH, Greece. He holds a Ph.D in Computer Networks (2022) and Diploma in Electrical and Computer Engineering (2016) from Democritus University of Thrace. He has been working in R&D projects for several years, both in academia and the industry, and has published several peer-reviewed papers and journal articles. His current research interests lie in the areas of cloud/edge computing and IoT, security and trust mechanisms, and networking protocols.

Giannis Ledakis (male) is a Senior Research Engineer, working as Team Leader in Computing Systems, Software and Services Research Group at UBITECH Ltd. He participated in commercial projects but also in European and National R&D programs, where he contributed from both technical and managerial perspectives. Through these projects he has collected valuable experience in many state-of-the-art technologies in various ICT fields, including Cloud Computing, Virtualization, Microservices, and Security. He graduated from the department of Computer Engineering and Informatics of the University of Patras.

Vassilis Tsaoussidis (male) is a professor at the Democritus University of Thrace, Greece and adjunct professor at Athena RC, holding degrees in Applied Mathematics (Aristotle University) and Computer Science (Ph.D. in computer science, Humboldt University, Berlin). Vassilis held faculty appointments at the Computer Science Department of SUNY Stony Brook, NY, US and at the College of Computer Science of Northeastern University, Boston, MA, US. Vassilis was/is associate editor in IEEE Transactions on Mobile Computing, Computer Networks, Ad hoc Networks and others. He was member of the board of Directors of the Hellenic Space Agency and member of the Board of Democritus University. Vassilis co-ordinated several EU projects such as FP-7 SPICE, FP-7 Space Data Routers, HORIZON UMOBILE, NGI projects such as Trusted Content Networking, and ESA projects such as Extending Internet into SPACE.