Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

ProTrust: A Probabilistic Trust Framework for Volunteer Cloud Computing

YOUSEF ALSENANI, * GARTH CROSBY (SENIOR MEMBER, IEEE), KHALED R. AHMED, AND ** TOMAS VELASCO ,

School of Computing, Southern Illinois University at Carbondale, IL, 62901 USA (e-mail: yalsenani, khaled.ahmed@siu.edu)
*The Department of Engineering Technology and Industrial Distribution Texas A&M University College Station, TX 77843 USA (e-mail: gvcrosby@tamu.edu)
**School of Applied Engineering and Technology, Southern Illinois University at Carbondale, IL, 62901 USA (e-mail: velasco@siu.edu)

Corresponding author: Yousef Alsenani (e-mail: yalsenani@siu.edu).

ABSTRACT With the exponential growth of large data produced by IoT applications and the need for low-cost computational resources, new paradigms such as volunteer cloud computing (VCC) have recently been introduced. In VCC, volunteers do not disclose resource information before joining the system. This leads to uncertainties about the level of trust in the system. The majority of available trust models are suitable for peer-to-peer (P2P) systems, which rely on direct and indirect interaction, and might cause memory consumption overhead concerns in large systems. To address this problem, this paper introduces ProTrust, a probabilistic framework that defines the trust of a host in VCC. We expand the concept of trust in VCC and develop two new metrics: (1) trustworthiness based on the priority of a task, named *loyalty*, and (2) trustworthiness affected by behavioral change. We first utilized a modified $Beta$ distribution function, and the behavior of resources are classified into different loyalty levels. Then, we present a behavior detection method to reflect recent changes in behavior. We evaluated ProTrust experimentally with a real workload trace and observed that the framework's estimation of the trust score improved by approximately 15% and its memory consumption decreased by more than 65% compared to existing methods.

INDEX TERMS Volunteer Cloud Computing; Volunteer Computing; Cloud Computing; Trust; Reputation; IoT.

I. INTRODUCTION

Currently, the Internet of things (IoT) refers to billions of devices connected to the Internet, which produce a huge amount of data. These data need sufficient computing resources for collecting, processing, and computing real-time data analytics. The migration of latency-sensitive IoT applications, data-intensive services such as big data analytics, and highly sensitive medical data to the public cloud environment has become less adaptable [1], [2]. Although researchers and scientists utilize public cloud service providers such as Amazon Web Services (AWS) and Microsoft Azure to conduct their research experiments, high costs remain a major concern. For example, in 2012, two computational chemistry companies built a 50,000-core supercomputer for a cancer research project with AWS that ran for three hours at a cost of $4,829 per hour [3]. These costs can often be restrictive for many universities.

Volunteer cloud computing (VCC) [4], [5] is evolving to meet these requirements through the utilization of spare resources of personal computers (referred to as volunteer hosts) owned by individuals and organizations. There are billions of personal computers (PCs) and devices being used around the world. These PCs and other resources in university laboratories are typically used during regular working hours and are idle at night and on the weekends. Most of these PCs are underutilized, which has given rise to the idea of volunteer cloud computing using underutilized PCs (i.e., volunteer hosts). A volunteer cloud computing approach that makes use of spare resources has the potential to complement the traditional cloud computing model [6].

Volunteer cloud computing has different characteristics than other existing platforms. One of the unique features is the ability to allow participants to take advantage of virtualization technology to donate their computing resources for distributed applications. This feature, however, leads to uncertainty about the level of trust between the task and

volunteer host. In volunteer computing, many issues can arise if the trust is simply assumed. Volunteer hosts willingly donate their computing resources for shared computing tasks; however, they do not disclose the details of their resources in terms of identity and availability. A lack of trustworthiness can result in undesirable activity. The behavior of the volunteer host can change over time for different reasons (e.g., volunteer's intent, malicious behavior, and age of the machine). Moreover, volunteer hosts have finite computing and memory capacity. Thus, mechanisms are needed to create a trusted environment. One approach is to use a probabilistic system capable of tracking the trustworthiness of each member. The behavior of each volunteer host could be captured and estimated by means of an effective memory solution.

Volunteer cloud computing uses a client/server architecture, where a dedicated management server evaluates the trust relationship with clients (i.e., the volunteer host in this case) through direct interaction. A lack of trust might cause poor decision making within the system. In contrast, the peer-to-peer (P2P) paradigm relies on direct interaction and a measure of witnesses' observations (reputation) of a node to determine trust [7]. Volunteer cloud computing needs robust and adaptive mechanisms to prevent data loss and reduce task failures on untrustworthy volunteer hosts [6]. The nature of IoT applications that operate within a volunteer cloud environment is notoriously diverse and may include batch-type applications (e.g., a shopping application) with low priority and more performance-sensitive tasks/data with high priority (e.g., a health application) [8]. These differences indicate a need to consider the quality of service (QoS) requirements to achieve a more optimal task assignment to volunteer hosts. For fine-grained trust computations, performance sensitivity levels must form part of the trust model.

To address these problems, this article introduces ProTrust, a probabilistic framework to determine the trust of volunteer hosts in volunteer cloud computing. ProTrust uses a probability distribution approach to define the trust of volunteer hosts. We implement a variant of a distribution function that is adaptive to the level of loyalty of the volunteer host behavior. Our new version of the *Beta* function is used as the probability density function (PDF). It requires only two parameters, which reduces memory consumption, especially given the resource constraints of volunteer hosts. The ProTrust process comprises three phases. In the first phase, the volunteer host's behavior is classified into different levels of loyalty. In the second phase, the level of trust the volunteer host exhibits towards the system is computed based on past behavior. Finally, a behavior detection method is used to detect any behavioral changes.

The contributions of this research are summarized as follows:

- We highlight the essential differences between volunteer cloud computing and existing platforms. Discussion of the unique characteristics of volunteer cloud computing can help in designing appropriate solutions for the unique problems.
- ProTrust is a new approach for estimating the trust of a volunteer host using a probabilistic framework that exploits our new version of the *beta* distribution. The Beta distribution is adapted with our first motivated solution known as *loyalty*.
- We use a behavior detection method to capture recent changes in behavior.
- Experimental data on the effectiveness of ProTrust are provided, and ProTrust is compared with other approaches.

The remainder of this paper is organized as follows: We discuss the characteristics of volunteer cloud computing in Section II. We discuss the motivation in Section III. Then, we review related work in Section IV. In Section V, we provide definitions and assumptions. We present ProTrust in detail in Section VI. We introduce the dataset in Section VII. Then, in Section VIII, we discuss the experimental results. We conclude in Section IX.

II. VOLUNTEER CLOUD COMPUTING

In this section, we highlight the essential differences between volunteer cloud computing and other existing platforms. Whether VCC offers an economical alternative to the cloud computing environment or not, the current dedicated cloud system may not be appropriate in some of the following IoT application scenarios:

- Dispersed data-intensive services (big data): This scenario requires migrating large amounts of data. Moving data onto the compute node that will process these data provides better performance and QoS.
- If a research group would like to execute a distributed task, a volunteer cloud model is an appropriate and affordable solution [9].

The high-level architecture of a volunteer cloud is depicted in Figure 1. Although the volunteer cloud shares many similarities with the traditional cloud and desktop grid computing models, there are several essential differences, as follows:

1) **High resource heterogeneity**: Most traditional models of cloud computing are employed on top of a homogeneous infrastructure, whereas volunteer cloud models run on highly heterogeneous hosts.
2) **Resource availability and volatility**: Traditional cloud applications run on dedicated infrastructure, whereas volunteer cloud models are volatile and have varying levels of volunteer host availability due to volunteers randomly joining and leaving the network.
3) **Trustworthiness of volunteer hosts**: In volunteer cloud computing, trust relationships between the volunteer hosts and the volunteer system are required.
4) **Diversity of workloads**: The volunteer cloud model aims to host a different type of application, whereas traditional computing, such as a desktop grid, usually targets and is suitable for CPU-intensive scientific applications.

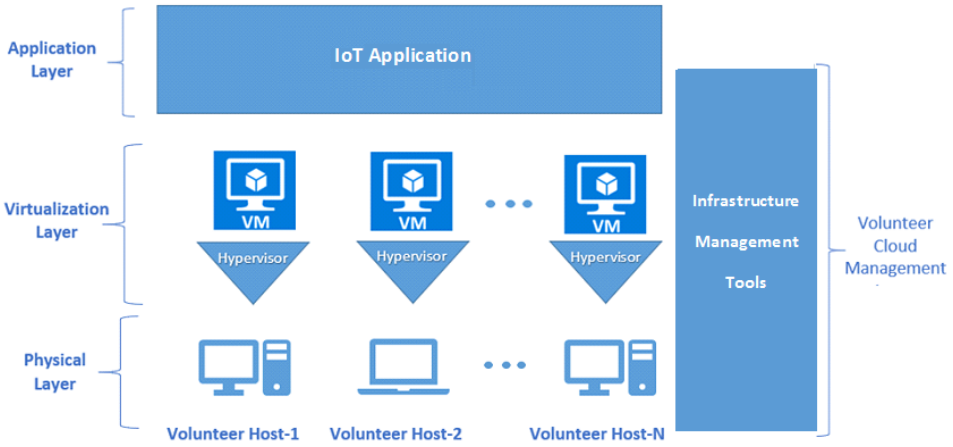


FIGURE 1: High-level Architecture of a Volunteer Cloud

The trustworthiness of volunteer hosts is complex; thus, straightforward application of the current methods to volunteer computing would not result in satisfactory performance.

III. MOTIVATION

In this section, we discuss the weakness of trust models in distributed systems, which motivates our proposed framework. Volunteer cloud computing (VCC), along with IoT application, is a new paradigm. Trust in this new paradigm is still obscure because of the nature of IoT applications and the characteristics of VCC. Although we cannot directly apply existing trust models, these models can help us to better understand trust in VCC. There are some limitations of current trust models that motivate us to build a suitable trust framework.

On the one side of the spectrum, in a peer-to-peer (P2P) desktop grid, the evaluation of node trust is based on direct interaction and refers to witnesses' observations (i.e., reputation) when there are few direct interactions between nodes [7]. Cloud computing, which relies on server/client architectures, considers only direct interaction between the server and clients to establish a trust relationship, as clients do not interact with each other. Therefore, few interactions raise a challenge in terms of the trust. Resource management in volunteer cloud computing should incorporate different levels of IoT application characteristics/priorities and volunteer host behavior to build an accurate trust framework. On the other side of the spectrum, monitoring the trust of resources may result in memory overhead in large-scale systems due to resource constraints in the volunteer cloud. Reducing space memory requirements ultimately improves the system performance in the volunteer cloud environment.

Several works evaluate trust resource models in P2P systems and assume that tasks are homogeneous in terms of priority [10], [11]. These models do not incorporate important task characteristics (such as priority), nor do they capture changes in behavior to make fine-grained trust decisions. Therefore, we observed several drawbacks:

- Current trust models evaluate the trust of resources by considering the entire historic data. However, this method can result in two issues: (1) behavior can change over time for different reasons (for example, it may exhibit a "bathtub curve"), and (2) storing all transactions requires considerable storage space. Thus, estimating trust based on the most recent behavior can provide an accurate estimation and reduce memory overhead by removing a portion of the past transactions.
- The following hypothetical sample should be considered. Let us assume that Host Machine A has an elevated rate of task failure compared to Host Machine B. As a result, Host Machine B will have a higher trust score assigned to it. However, when Host Machine B handles higher priority and performance-sensitive tasks and Host Machine A handles batch (lesser priority) tasks, the existing model allocates a better trust score to Machine A because Host Machine A has an increased chance to complete more batch-type jobs than Host Machine B. Thus, the failure rate cannot be relied upon as an independent trust measure when the number and type of tasks are not taken into account.
- The current trust models assign an identical success probability to all tasks based on the individual trust of the resource. However, untrusted resources with high-priority tasks should have a reduced probability of success.
- The same trust score for these trust models is given to all volunteer hosts with the same success rate, regardless of the characteristics of the tasks that have been successfully completed. Thus, when a volunteer host does not successfully complete a range of sensitive and high-priority tasks, it could still have an inflated trust score. As such, volunteer hosts can negatively affect trust if sensitive or high-priority tasks fail while on their resources.

Volunteer cloud computing relies on the direct interaction between the server and volunteer host to establish a trust relationship. Lack of information, fewer interactions involving the host, or behavioral changes over time may lead to poor system decisions about this volunteer host. These problems inspired us to develop loyalty and behavior detection method as a solution mechanism to assess the relationship between resource management and volunteer hosts.

IV. RELATED WORKS

Trust domain has long been of interest to various disciplines such as computer science, social science, and business. Trust is a critical and important aspect that assures a good relationship between entities. In this paper, we review related literature on volunteer computing and distributed systems. Several volunteer computing approaches have proposed security and trust models to overcome potential malicious activity. Different models have used trust and reputation models to evaluate resources [12]. Trust between individual entities is facilitated by reputation, and reputation systems with different dimensions (e.g., history, context, collection, representation, and aggregation) are discussed in the literature [13]. Reputation- and trust-based models are used in different domains such as online auctions (e.g., eBay), P2P systems [14], volunteer clouds [11], grid systems [15], social networks [16], the mobile social cloud [17], and WSN [18]–[20]. These models in different domains evaluate trust based on performance (i.e., past transactions/behavior). First, we discuss the state of the art in trust management in a distributed system. Second, we discuss papers that exploit a similar probability distribution function and show how our function differs. Third, we present some well-known security/trust techniques in volunteer computing. Finally, we briefly discuss our research justification.

Trust management can be classified into two categories: single-sourced and multi-sourced trust models [21]. In single-sourced trust, trust is evaluated by policies such as QoS parameters, audit assessment, and accountability [22]. Whereas some work has used trusted third-party-supported applications that allow the user to specify their security preferences [23], the authors in [24] proposed hard and soft trust, where hard trust is a factor of security mechanisms and soft trust comes from previous experience. Moreover, trust can be evaluated by policy-based models with multicloud support; for example, work concerning [25] the overlay network and compliance management has been conducted to evaluate the trust in multidata centers. A recommendation can be one source of trust. For instance, the trust of a resource can be evaluated based on peer feedback [26]. Zhao [27] proposed a privacy-preserving reputation model for mobile crowdsensing (MS) that integrated new technology, such as the blockchain and edge computing. Another method to evaluate trust is by evidence. For example, work in [28] evaluated trust by predicting the behavior of resources for resource selection. In the second category, multi-sourced trust models, trust is evaluated by supportive evidence [29]. In [30], the authors evaluated trust and reputation by means of D-S evidence theory and a sliding window. Moreover, Cloud-Trust [31] used different factors, including security, reliability, and availability, to evaluate trust, whereas other studies have used machine learning models to predict trust [32]. In Multi-Cloud, the authors in [33], [34] proposed a broker concept for cloud provider selection. The trust model can be evaluated in terms of the overall policy and recommendations. The author in [35] combined measures of trust and competence for interaction risk estimation. Finally, there are some trust models with federation support. Mashayekhy [36] presented a trust-aware model in the federation cloud that relies on both the trust and reputation of the cloud provider.

Now, we discuss research on trust in different domains of distributed systems that exploit the use of probability distribution functions. For open-ended cloud environments, the authors presented a reputation-based trust model in [11]. The proposed work calculated the reputation score of the resource and assigned tasks to nodes based on reputation score by applying a probabilistic Beta model, exploiting the expected value as

$$E(x) = \frac{\alpha}{\alpha + \beta} \tag{1}$$

where α is the number of successful tasks and β is the number of failed tasks. There are three different node selection strategies: (1) random (R), a node is selected regardless of the reputation score; (2) reputation-based (RB), the node with the highest reputation value is chosen; and (3) probabilistic reputation-based (PRB), the node is selected randomly using a probability distribution over the available nodes. The PRB strategy gives nodes with low reputation value a chance to improve their score. Even though we exploit the same probabilistic distribution function, the behavior in ProTrust employs different levels of loyalty based on the task characteristics

The Dirichlet distribution has also been exploited in the trust domain [37]–[39]. The Dirichlet distribution is a multivariate version of the Beta distribution. In general, the Dirichlet distribution is used to model events with more than two distinct results (e.g., success, failure, missing, etc.), unlike the Beta distribution, which takes only two distinct results (e.g., success, failure). The Dirichlet distribution with parameters $\alpha_i{}_{i=1}^{k}$ is a discrete probability distribution of α_i positive results with a random variable x_i in a sequence of $\sum_{j=1}^{k} \alpha_j$ independent trials with a probability of:

$$\alpha_i / \sum_{j=1}^{k} \alpha_j \tag{2}$$

where $K > 2$ in the Dirichlet distribution and $K = 2$ in the Beta distribution.

The authors in [26] proposed a trust protocol for the IoT environment. The authors computed trust using direct trust characteristics such as user feedback and indirect trust characteristics such as recommendations from other peers. Recommendation were evaluated based on social similar-

ity, including social contact, friendship, and community of interest. Further, the authors combined direct and indirect interactions for trust evaluation and adjusted the node weight values by means of an adaptive filtering model. Unlike [26], we do not exploit the social relationships between volunteer hosts to maintain trust since it might be difficult to acquire social information from volunteers because of privacy issues. The authors use a Beta system to estimate trust, and they use the following equations to update the shape parameters α, and β:

$$\begin{aligned} \alpha_{x,i} &= e^{-\varphi \Delta t} \cdot \alpha_{x,i}^{old} + f_{x,i} \\ \beta_{x,i} &= e^{-\varphi \Delta t} \cdot \beta_{x,i}^{old} + 1 - f_{x,i} \end{aligned} \tag{3}$$

where $f_{x,i}$ and $1 - f_{x,i}$ represent positive and negative experiences, respectively, and $e^{-\varphi \Delta t}$ is the decay factor of the contribution of old observations to the model. The authors use the similarity in social relations between peers as prior information and assign the initial values of $\alpha_{x,i}$ and $\beta_{x,i}$ to these similarity measures. In ProTrust, we set average values as prior knowledge of these shape parameters obtained from volunteer host outcomes.

For mobile cloud computing (MCC), a reputation, trust-aware, and privacy prevention schema is proposed in [40]. Both reputation and trust are modeled by a Beta distribution, and the authors present anonymous secure-shell ciphertext policy attribute-based encryption (AS-CABE) to manage privacy on the platform. The authors estimate the trust of the contributor and system user simultaneously. In their design, the data requester is responsible for measuring the quality of service. Thus, trust is evaluated based on feedback from different data requesters, the historical performance, and age, as

$$R = \sum_{j=1}^{m} \sum_{i=1}^{n} \gamma_j \alpha_i R_i^T \tag{4}$$

where R_i^T represents the rate assigned by the data requester, n is the history parameter, m is a number of tasks, α_i is a weight based on the history of task completion, and γ_j represents the maintained history (e.g., age of feedback). In contrast to this approach, the behavior in our work considers different levels of loyalty based on task characteristics.

For social IoT, Lin and Dong in [41] proposed a trust framework that considered various concepts, such as 1) mutuality of trustor and trustee, 2) the inferential transfer of trust, 3) transitivity of trust, 4) trustworthiness update, and 5) trustworthiness affected by dynamic environment. They aimed to overcome limitations in current social trust IoT models by accounting for the above five aspects.

For P2P networking, the authors in [14] proposed EigenTrust, a trust and reputation model that assists in making trust decision between nodes. Each node is granted a global trust score based on past performance. For WSNs, [19] presented location-aware and trust-based solutions to detect and isolate compromised nodes. They employed a Beta distribution to model trust and reputation in the network. For a social volunteer cloud, the authors presented a social compute cloud, that is, a cloud approach that uses social relationships (like friendships on Facebook) to define the trust between nodes. Several attempts have been made to detect unfair ratings and malicious nodes; however, these models require large amounts of information or prior knowledge for precise evaluation [42]–[44]. The authors in [45] proposed a reputation framework for identifying denial of traffic service to establish trust in VANETs

ProTrust shares a similar concept to Moyano *et al* [46] in evaluating trust based on the priority of the task. The authors evaluate the trust between entities in the system by the task order. First, the model orders the tasks by computing a risk assessment based on all entities for all tasks. Second, the model calculates the entities divergence (ED) and trust incremental values (TIV). Finally, the authors propose order-dependent trust evaluation to compute the trust between the entities for each task. However, in ProTrust the volunteer host is penalized or awarded (e.g., trust computation) based on the task outcomes, and behavioral changes are incorporated into the trust computation.

Other techniques in volunteer systems provide secure and trusted environments to limit malicious activity. For example, in result falsification, each task is submitted redundantly to a number of hosts, and the management server gives credibility to nodes that return the correct results. Different schema are used in result falsification, such as spot-checking and credibility-based voting An m-first/majority voting model is used in SETI@home and BOINC [47], [48] to collate the matching results in order to complete a task. Furthermore, a credibility-based approach to voting is followed in [49] to assign credibility and value to each individual worker node. ECJMAX was proposed in [50] as a specific model based on locating the minimal number of nodes according to their individual credibility. The authentication technique overcomes denial of service (DoS) attacks and malicious executable distribution by requiring users to use new key pairs periodically and limiting the size of tasks to a given threshold. Moreover, sandboxing and virtualization technologies can be used to deploy guest tasks in controlled environments to prevent native and in-native users from attacks [51]. In IoT search [52], access control mechanisms are discussed, where the characteristic of IoT search such as dynamic environment and massive data are highlighted.

We conclude this section by highlighting the issues we observed in the related work. Evaluating trust without considering the task priority to make fine-grained trust decisions and behavior changes at the volunteer host level are some of the limitations in the literature. First, we highlight two aspects of the concept of trust in VCC, (1) the trustworthiness by priority of the task (2) and trustworthiness affected by behavioral changes; then, we propose loyalty and behavior detection mechanisms to distinguish between trusted and untrusted volunteer hosts and capture recent changes in be-

havior. We implemented enhanced probability distributions that are adaptive to heterogeneity and different task priorities. Our adaptation mechanism differs from the original Beta function insofar as it addresses the drawback discussed above by means of a loyalty mechanism and provides a trusted volunteer cloud infrastructure. Age or time of interaction factor (e.g., relying on recent outcomes observed from peers) alone is not sufficient in the scenario of dynamic and constrained resources. Therefore, this work proposes the ProTrust framework, which considers how trustworthiness is affected by behavioral change.

V. PRELIMINARIES

In this section, we provide definitions, assumptions, and proposition that are used throughout the paper.

A. DEFINITIONS AND ASSUMPTIONS

Definition. Trust, the level of trust a volunteer host exhibits towards the system, is computed based on past behavior.

Assumptions. We assume that all volunteer hosts in the volunteer cloud function properly, and all task failures result from untrusted volunteer hosts. We also assume that all management servers are dedicated and trusted.

B. PROPOSITION

Proposition 1. Assume that server S has assigned task to volunteer host r. Then, the outcome of the task for volunteer host that S can observe is summarized by a binary variable x, defined as:

$$x = \begin{cases} 1 & \text{if } \textit{Task} \text{ succeeded} \\ 0 & \text{if } \textit{Task} \text{ failed} \end{cases} \tag{5}$$

The history of task outcome is recorded, and the value of α is the number of successful task and β is the number of unsuccessful task. The probability of a volunteer host r completing the next task successfully is defined as τ_r. Then, the posterior distribution of successful task of r follows Bayes' theorem, and the posterior distribution can be written as

$$P(\tau|x_1, ..., x_n) = \frac{P(x_1, ..., x_n|\tau)\pi(\tau)}{P(x_1, ..., x_n)} \tag{6}$$

While Bayes' estimator is:

$$E[\tau|x_1, ..., x_n] = \int_0^1 \tau p(\tau|x_1, ..., x_n) d\tau \tag{7}$$

Proof. where $x_1, ..., x_n \sim \text{Bernoulli}(\pi(\tau))$. Suppose we take the uniform distribution $\pi(\tau) = 1$ as a prior, which leads to Beta (1,1). By Bayes' theorem, the posterior is

$$P(\tau|x_1, ..., x_n) \propto \pi(\tau)\mathcal{L}(\tau) = \tau^{S_n+1-1}(1-\tau)^{n-S_n+1-1} \tag{8}$$

where $S_n = \sum_{i=1}^{n} x_i$ is the number of successes. Recall that a random variable τ on the interval (0, 1) follows a Beta distribution with parameters α, β if its density

$$\pi_{\alpha,\beta}(\tau) = \frac{\Gamma(\alpha)\,\Gamma(\beta)}{\Gamma(\alpha+\beta)}\tau^{\alpha-1}(1-\tau)^{\beta-1} \tag{9}$$

Now, the posterior distribution for τ is a Beta distribution with parameters $S_n + 1$ and $n - S_n + 1$.

Therefore, the Bayes' posterior estimator is the expected value for the beta function and can be defined as:

$$\tau_r = E[Beta(\tau|\alpha+1, \beta+1)] = \frac{\alpha+1}{\alpha+\beta+2} \tag{10}$$

where we can use this distribution function to predict the probability in the future, which is the trust in our case.

C. BETA DISTRIBUTION

The beta distribution has the pdf $prob(x|\alpha, \beta)$ for a value x on the interval (0,1) and can be expressed using the gamma function, Γ, as follows:

$$Beta(\alpha, \beta): \ prob(x|\alpha, \beta) = \frac{\Gamma(\alpha)\,\Gamma(\beta)}{\Gamma(\alpha+\beta)}x^{\alpha-1}(1-x)^{\beta-1}, \tag{11}$$

where $0 \leqslant x \leqslant 1, \alpha > 0$, and $\beta > 0$, with the restriction that the probability variable $x \neq 0$ if $\alpha < 1$. The expected value for the beta distribution is defined as follows:

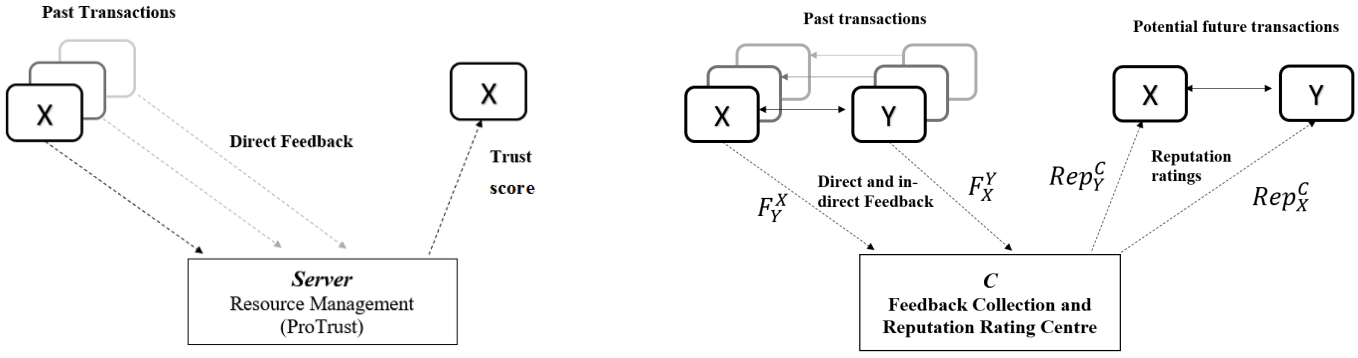
$$E(x) = \frac{\alpha}{\alpha+\beta}, \tag{12}$$

where x is a probability variable.

The proposed ProTrust framework evaluates the trustworthiness of resources through direct interactions between volunteer hosts and the dedicated management server. This approach is different from the traditional trust framework, where reputation and trustworthiness are evaluated through direct interaction between two volunteer hosts and by indirect interactions, known as the reputation of the volunteer host, which is evaluated by the community. Figures 2a, 2b illustrate the ProTrust framework and the traditional trust framework. In ProTrust, the interaction or feedback between volunteer host X and the server is direct. The management server hears about X from the only X. However, in the traditional trust framework, the reputation center (assume that each volunteer host has a reputation center) evaluates X in terms of direct and indirect interactions. Thus, the reputation center can hear about X from X itself and also from other volunteer hosts in the network, such as volunteer host Y or volunteer host Z, as there have been interactions between volunteer hosts X and Y and between volunteer hosts X and Z. Thus, volunteer host Y or Z can share X's outcomes with the network or community.

VI. PROTRUST FRAMEWORK

In this section, we discuss the notation of ProTrust. Then, we discuss the new concepts of (1) trustworthiness in terms of task priority, named *loyalty*, and (2) trustworthiness affected by behavioral change. Finally, we present the ProTrust algorithm.



(a) ProTrust Framework

(b) Traditional Reputation Framework

FIGURE 2: ProTrust and Traditional Trust Frameworks

A. PROTRUST NOTATION

A volunteer cloud computing scenario assumes a client/server architecture (e.g., the client is a volunteer host, and the server is a dedicated management host). A volunteer cloud platform is a typical distributed system with distributed service. When a cloud user requests service from the platform, the management server is responsible for distributing this service as a task or set of tasks to a single host or a set of volunteer hosts r_j, where $j \in [1, 2, \ldots, p]$. Volunteer host selection is based on a trust score that is calculated based on the volunteer hosts' performance history. Assume that server S has assigned task to volunteer host r. Then, the task outcome for volunteer host that S can observe is summarized by a binary variable x, defined in equation (5).

At any time t, the history of observed task outcomes for r is recorded in the 2-tuple $\Re^t = (\gamma_s(x)^t, \gamma_f(x)^t)$, where the value of the function $\gamma_s(x)^t$ is the number of successful task, and the value of $\gamma_f(x)^t$ is the number of unsuccessful task. The obligation of a volunteer host r to successfully complete tasks is controlled by its behavior [53]. The behavior of r, denoted as B_r, is modeled as the probability that $x = 1$. In other words, B_r specifies the probability that r will commit its task during an interaction, which is defined as follows:

$$B_r = p[x = 1], \text{where } B_r \in [0, 1].$$

After many interactions, we can monitor the behavior of host r by calculating B_r. In statistics, we cannot assume that we have all the information and complete knowledge. However, we can exploit the expectation. Thus, we define trust τ_r at time t as the expected value of B_r given the number of task outcomes x:

$$\tau_r = E[B_r | x^{1:t}],$$

where $x^{1:t}$ is the set of task outcomes from time 1 to the assessment time t. The expected value of a continuous random variable depends on the probability density function utilized to shape the probability that the variable will have a certain value [54]. To this end, the ProTrust framework estimates the trust of a volunteer host using the Beta distribution [55] presented in Section V-C. The Beta system uses the standard $beta$ distribution, which works well with prior expectations. In our case, when a new volunteer host joins the system, not much information is disclosed. The $beta$ distribution is a two-parameter distribution that is commonly used in trust models. Several platforms and domains, such as peer-to-peer (P2P) networks, wireless sensor networks (WSNs), and grid and cloud computing [15], [19], also exploit $beta$ distributions to determine the probability of successfully executed tasks on a specific volunteer host. This distribution is adaptable and has a robust foundation in probability theory [55]. While alternative methods, for instance, machine learning, may deliver more accurate results, there are a number of known issues, such as the cold start problem and elongated prediction times due to the small number of samples obtained from newer volunteer hosts.

B. TRUSTWORTHINESS BY PRIORITY OF TASK

Limitation 1. *Trustworthiness is evaluated irrespective of task priority*

Existing trust models assign better trust scores to hosts that complete lower-priority less-sensitive tasks (more batch tasks) within short periods of time. Additionally, a lower trust score is given if the volunteer host completes more high-priority or performance-sensitive tasks that require a greater amount of time to complete. Moreover, an identical probability of success is given by these models to volunteer hosts based on the resource's trust score. However, untrusted resources conducting highly sensitive tasks should have a reduced probability of success. Volunteer hosts with the same rate of success are granted an identical trust value, regardless of the individual characteristics of the successfully completed tasks. A failure across a range of sensitive and high-priority tasks would limit the effectiveness of the trust score to distinguish between untrusted and trusted volunteer hosts. For instance, *in law, regardless of the classification of a theft, be it a grand felony, petty or grand, the determining factor for the outcome is decided upon by the value of the stolen property*. To address these limitations, we propose loyalty,

which associates outcomes with different levels of priority. For instance, we group the visible results, such as the failure and success rates, with regards to various characteristics: this could be the sensitivity of a task or IoT latency (i.e., high priority/sensitive, medium priority/sensitive, and low priority/batch). Thus, failure or success in a range of latency-sensitive or high-priority tasks would dominate the outcome and impact trust value. For example, consider a researcher who needs to analyze experimental data (e.g., assume his research is not published yet, and the privacy of this data is mandatory) using affordable computing resources, where there is a number of volunteer hosts available to host tasks in his/her university. It would be beneficial for resource management to provide the trust scores of these volunteer resources that are able to host this sensitive data. Thus, loyalty is a mechanism that clusters direct interactions into different levels. Below, we show how to rank the task outcomes from volunteer hosts with respect to different priorities. In equation (13) we show how to compute t_{high}^{succ} for the successful high-priority task outcomes.

$$t_{high}^{succ} = \sum_{i=1}^{N} x_i * \omega_h, \tag{13}$$

Where x_i are the successful high-priority task outcomes and ω_h are constant weights. Where N is the number of successful high-priority task. We follow the same approach for medium- and low-priority task outcomes, as follows:

$$t_{med}^{succ} = \sum_{i=1}^{N} x_i * \omega_m, \tag{14}$$

$$t_{low}^{succ} = \sum_{i=1}^{N} x_i * \omega_l, \tag{15}$$

where t_{med}^{succ} and t_{low}^{succ} are the total numbers of successful medium- and low-priority task outcomes, respectively, and ω_m and ω_l are constant weights for medium- and low-priority task outcomes, such that $0 \leqslant \omega_h + \omega_m + \omega_l \leqslant 1$ and $\omega_h > \omega_m > \omega_l$. After observing a certain amount of behavior, the loyalty $\gamma_s(x)$ is calculated as the total number of successful high-, medium-, and low-priority task by combining equations (14–16), as follows:

$$\gamma_s(x) = t_{high}^{succ} + t_{med}^{succ} + t_{low}^{succ}. \tag{16}$$

To compute $\gamma_f(x)$, which represents the negative feedback for unsuccessful task, we follow the same procedure we used for $\gamma_s(x)$:

$$t_{high}^{unsucc} = \sum_{i=1}^{U} x_i * \delta_h, \tag{17}$$

where t_{high}^{unsucc} is the total number of unsuccessful high-priority task outcomes, x_i represents unsuccessful task outcomes, δ_h is a constant weight, and U is the number of unsuccessful high-priority task outcomes. We use the same technique for medium- and low-priority task, as follows:

$$t_{med}^{unsucc} = \sum_{i=1}^{U} x_i * \delta_m, \tag{18}$$

$$t_{low}^{unsucc} = \sum_{i=1}^{U} x_i * \delta_l, \tag{19}$$

where t_{med}^{unsucc} and t_{med}^{unsucc} are the total numbers of unsuccessful medium- and low-priority task outcomes, respectively, and δ_m and δ_l are constant weights for medium- and low-priority tasks, such that $0 \leqslant \delta_h + \delta_m + \delta_l \leqslant 1$ and $\delta_h > \delta_m > \delta_l$. After observing a certain amount of behavior, the loyalty $\gamma_f(x)$ is calculated as the total number of unsuccessful high-, medium-, and low-priority tasks outcomes by combining equations (19–21), as follows:

$$\gamma_f(x) = t_{high}^{unsucc} + t_{med}^{unsucc} + t_{low}^{unsucc}. \tag{20}$$

C. UPDATING THE SHAPE PARAMETER

The ProTrust framework depends on feedback received from volunteer hosts, which can be successful or unsuccessful execution of tasks. In VI-B, we discuss how the behavior of volunteer hosts can be captured. To compute a volunteer host's trustworthiness τ_r, we must find the shape parameter for the beta function. We define the parameter α as the number of successful tasks and β as the number of unsuccessful tasks executed on a given volunteer host. Then, we define the PDF for the beta function as follows:

$$Beta(\alpha, \beta) = \frac{\Gamma(\alpha)\,\Gamma(\beta)}{\Gamma(\alpha+\beta)} x^{\alpha-1}(1-x)^{\beta-1}$$

where $0 \leqslant x \leqslant 1, \alpha > 0$, and $\beta > 0$.

The loyalty we have implemented in this technique is adaptive to heterogeneity and different priorities of tasks. We use α to represent the number of positive and successful outcomes, which depends on the level of loyalty $\gamma_s(x)$ for successful outcomes for volunteer host r_j, as follows:

$$\alpha_j = \gamma_{sj}(x)^{1:t} + \frac{1}{p-1} \sum_{i=1, i\neq j}^{p} \alpha_i \tag{21}$$

where $\gamma_{sj}(x)^{1:t}$ is the successful tasks from time 1 to assessment time t for volunteer host r_j. The second part of the above equation calculates the average value of all other volunteer hosts' α and sets this value as prior knowledge, where p is the number of volunteer hosts. Unlike works in literature where α, β are set to 1 since no prior knowledge exists, we believe that it is fair for a new volunteer host that just joins the system to start with the average trust value.

We follow the same technique for negative volunteer host feedback. We define β as the amount of negative feedback, which depends on the level of loyalty $\gamma_{fj}(x)$ for unsuccessful outcomes for volunteer host r_j, as follows:

$$\beta_j = \gamma_{fj}(x)^{1:t} + \frac{1}{p-1}\sum_{i=1, i\neq j}^{p}\beta_i \tag{22}$$

D. MODELING TRUST

Next, we estimate the trustworthiness of volunteer host r_j with the expected value of a random variable according to the beta distribution $Beta(\alpha,\beta)$:

$$\tau_{r_j} = E[B_r|\alpha_j, \beta_j] = \frac{\alpha_j}{\alpha_j + \beta_j}. \tag{23}$$

This yields a trust metric in the range [0, 1].

E. TRUSTWORTHINESS AFFECTED BY BEHAVIORAL CHANGE

Limitation 2. *The update of trustworthiness does not consider behavioral change of the volunteer host.*

ProTrust considers the influence of past task performance and behavior changes over time in the dynamic and constrained infrastructure. Therefore, we propose a behavior detection approach that utilizes a method for detecting change points. Behavior detection is beneficial for two reasons. First, some volunteer hosts aim to improve their trust score (e.g., those who had a bad start) and some volunteer hosts may have recently begun to act maliciously. Behavior detection aims to minimize task failures by not assigning tasks to volunteer hosts whose behavior has recently changed for the worse. Second, behavior detection is a memory-efficient schema, which is desirable because volunteer hosts have limited computation power and memory space, that relies on only recent outcomes and ignores old outcomes. Many trust models [26] consider a time factor or age in the trust evaluation and use the most recent outcomes observed by peers. However, it is not a sufficient factor in dynamic and constrained resource scenarios. Behavior detection aims to find the optimal change point, which reflects the time at which the behavior is changed. There are different detection methods to capture any malicious activities or requests such as machine learning [56]–[58]. In this paper, to detect a change in behavior, we use the pruned exact linear time (PELT) method to search for change points [59]. In statistical analysis, a change point is an identified point of time within a dataset such that the statistical properties before and after this point differ. The PELT algorithm is a pruning step within the dynamic program. This mechanism has an appropriate computational cost and produces a more accurate segmentation than binary segmentation (BS).

For example, for the set of the tasks outcomes $x_1, \ldots, x_n$, if we assume that a change point occurs at z, then $x_1, \ldots, x_z$ differs from $x_{z+1}, \ldots, x_n$. In real-life scenarios, we might have multiple change points. Therefore, we define κ as the number of change points at positions $z_{1:\kappa} = (z_1, \ldots, z_\kappa)$, where $z_0 = 0$ and $z_{\kappa+1} = n$. The most common approach to find multiple change points is:

$$\min_{\kappa, \boldsymbol{z}} \left\{ \sum_{i=1}^{\kappa+1} \left[-\ell(x_{z_{i-1}:z_i}) \right] + \lambda f(k) \right\}. \tag{24}$$

This can be viewed as a special case of penalized likelihood. The aim is to maximize the (log-)likelihood over the number and position of change points, subject to a penalty that depends on the number of change points. The penalty is imposed to avoid overfitting. The formula can be cast in terms of minimizing a function of k and z of the following form:

$$CPT = \sum_{i=1}^{k+1} \left[\mathcal{C}(y_{(z_{i-1}+1):z_i}) \right] + \lambda f(k), \tag{25}$$

where $\lambda f(k)$ is a penalty to protect against overfitting, such as a threshold cost or a multiple change-point version, and $\mathcal{C}$ is a function of cost for a segment. There is also a penalty term that depends on the number of change points. After finding the most optimal change point (assume this change point is z), our trust value considers the history of observations after this change point—in other words, we rely on recent outcomes from the resource. We calculate the recent loyalty function $\gamma_s(x)^{z:t}, \gamma_f(x)^{z:t}$ to estimate the new trust based on recent transactions or outcomes (i.e., from time z to time t).

F. PROTRUST ALGORITHMS

In this section, we discuss the trust computation and behavior detection algorithms of ProTrust. These algorithms run on the management server after collecting the historical data from volunteer hosts.

Algorithm 1 for the trust computation is presented below. ProTrust takes the task outcomes for certain period of time from T_s to T_e where T_s is start time and T_e end time as input and estimates the trust score for each volunteer host. First, we check whether the volunteer host is already registered (line 1). In line 2, we check whether the tasks succeeded or failed. If the task succeeded, we update α discussed in equation (23). Otherwise, we update β discussed in equation (24). Then, we estimate the trust τ_{rj} after updating the shape parameters (line 8). Finally (line 11), the value represents a neutral rating for new volunteer hosts, where the shape parameters are the average values obtained from other volunteer hosts. The complexity of this algorithm is $O(p)$, where p is the number of volunteer host.

Algorithm 2 is used for behavior detection. The algorithm takes as input the task outcomes for certain period of time from T_s to T_e where T_s is start time and T_e end time. Then, it estimates the trust score after finding the optimal change point for the behavior. First, the detection method takes all observed outcomes and finds the time z of the behavior change point (lines 1–3). Then, the shape parameters α_j and β_j are updated by computing $\gamma_{sj}(x)^{z:t}$ and $\gamma_{fj}(x)^{z:t}$ for outcomes observed after time z (lines 4–5). Finally, the new trust value τ_{rj} is estimated (line 6).

Algorithm 1: Trust Computation

Input : Task outcomes for certain period of time from T_s to T_e
Output: τ_{r_j}

$Host = [r_1, r_2, ..., r_p]$
if *($r_j \in Host$)* **then**
  if *(task success)* **then**
    $\alpha_j \leftarrow \gamma_{sj}(x)^{1:t} + \frac{1}{p-1}\sum_{i=1,i\neq j}^{p}\alpha_i$
  end
  else
    $\beta_j \leftarrow \gamma_{fj}(x)^{1:t} + \frac{1}{p-1}\sum_{i=1,i\neq j}^{p}\beta_i$
  end
  $\tau_{rj} = \frac{\alpha_j}{\alpha_j+\beta_j}$
end
else
  $\tau_{r_j} \leftarrow \frac{\frac{1}{p}\sum_{i=1}^{p}\alpha_i}{\frac{1}{p}\sum_{i=1}^{p}\alpha_i + \frac{1}{p}\sum_{i=1}^{p}\beta_i}$
end

Algorithm 2: Behavior Detection

Input : Task outcomes for certain period of time from T_s to T_e
Output: τ_{r_j}

foreach $z = 1$ *to* $z = t$ **do**
  $cpt_z = \sum_{i=1}^{k+1}\left[\mathcal{C}(y_{(z_{i-1}+1):z_i})\right] + \lambda f(k)$
end
$\alpha_j \leftarrow \gamma_{sj}(x)^{z:t} + \frac{1}{p-1}\sum_{i=1,i\neq j}^{p-1}\alpha_i$
$\beta_j \leftarrow \gamma_{fj}(x)^{z:t} + \frac{1}{p-1}\sum_{i=1,i\neq j}^{p-1}\beta_i$
$\tau_{r_j} = \frac{\alpha_j}{\alpha_j+\beta_j}$

VII. DATASET TRACE

A data center cluster trace that was collected across a 29-day period in May 2011 was released by Google Inc. They also released a document titled 'Google cluster-usage traces: format+schema' that depicts in great detail the format, semantics, format, and schema of the trace [60]. This dataset includes host attributes, host events, constraints, tasks, jobs, and resource utilization information. The trace details jobs, and each job is composed of multiple tasks. The dataset includes information about approximately 26 million tasks as part of 672,074 jobs run on over 12,500 physical servers.

The tasks and jobs detailed within this cluster trace have specific event types that represent their status throughout their life cycles. The task and job status can be finished, killed, evicted, or failed. The task priorities are also classified into three categories; high, production, and low. The Google data center has a heterogeneous infrastructure, and there are ten different types of host machines. For the purposes of our study, we selected 100 physical hosts that present a large volume of failure patterns and behaviors. The highest failure rates for tasks throughout the life cycle of the data utilization trace was found in these physical hosts. The information contained within this trace has already been used by a number of papers on volunteer cloud and reliability assessment models to leverage real-world workload trace scenarios [61]–[63]. To the best of our knowledge, Google trace is the best real-world workload trace that includes our main contribution—task priority—that can be used to evaluate and validate our work.

VIII. PROTRUST EVALUATION RESULTS

In this section, the ProTrust approach is validated utilizing the Google cluster usage trace. Here, we compare the quality of ProTrust to that of the traditional trust framework [11], [55] and the Dirichlet distribution-based trust (DDMT) framework [38], [39].

A. EXPERIMENTAL SETUP

Our experiment was conducted on a Fujitsu computer with an intel(R) Core(TM) i7-4712MQ CPU @ 2.30 GHz, 2301 Mhz, and 8.0 GB memory. We used various software to conduct this research. To transform the dataset into the desired structure, we used SQL Server 2018 Express for data processing, and we used RStudio for change point behavior detection. As discussed in Section VII, the dataset consists of host attributes, host events, constraints, task events and characteristics, job events, and resource utilization information.

For the first introduced concept of loyalty (e.g., different task priorities), each task is assigned a positive number, where 0 represents the lowest priority (least important) and higher numbers represent higher priority. We retrieve the task events such as failure and success which represent the failure and success task, task characteristics such as task priority, and the associated hosts that performed these tasks. A sample of the data is shown in Table 1. The first column (i.e., Timestamp) refers to the time of the outcome. The second column (i.e., Host id) refers to the host identity. The third, fourth, and fifth columns (i.e., Success_H, Success_M, and Success_L) refer to the total numbers of successful high-, medium-, and low-priority tasks. The sixth, seventh, and eighth columns (i.e., Fail_H, Fail_M, and Fail_L) refer to the total numbers of unsuccessful high-, medium-, and low-priority tasks. We selected 100 hosts that present a large volume of failure patterns and behavior in their task performance.

For the second metric, behavioral change, which was discussed in Section VI-E, we simply pass the data discussed above to the change behavior method to identify the optimal change point. We focus on identifying changes *in the mean*, and we set $2 * log(datalength)$ for the penalty value, where $datalength$ is the length of data that are the outcomes for volunteer hosts. Then, we calculate the trust by exploiting the Beta distribution discussed in Section VI-B. Finally, we discuss the modeling and validation phase in detail in Section VIII-B.

TABLE 1: A Sample of aggregated data

Timestamp	Host id	Success_H	Success_M	Success_L	Fail_H	Fail_M	Fail_L
600000000	422055	1	0	2	0	0	0
900000000	603128	0	0	1	3	0	0
1200000000	907849	1	2	0	0	0	0

B. VALIDATING THE PROTRUST FRAMEWORK

The dataset consists of 29 days of data, and we divided the data into two parts. The first part, the modeling data (from days 1 to 27), is used to estimate the trust for a volunteer host. The second part (days 27 to day 29) is used to validate the trust. The validation considers the final successes and failures of tasks observed from day 27 to day 29 and is used to compare or validate the results of these estimations. As a final figure, we utilized the first 27-day period, which covers approximately 640 hours of the Google trace data, to effectively model ProTrust. Most of the volunteer hosts had changes in behavior; therefore, each volunteer host had a different input (e.g., we use data from days 15 to 27 for some volunteer hosts). Then, to validate the accuracy of ProTrust, we calculated the actual failure and success rates observed in the next 5 hours, i.e., hours 641–645 (next five hours–short prediction), and we calculated the actual failure and success rates of tasks observed on day 28 (next day prediction) and on days 28 and 29 (next 2 days–long prediction). We used the failure and success rates of tasks during these last two days to validate the estimation accuracy of the compared methods. We designate the different failure and success rates λ_i and ψ_i as follows:

- λ_{5hour} and ψ_{5hour} are the actual failure and success rates observed during the 5 hours after hour 640;
- λ_{1day} and ψ_{1day} are the actual failure and success rates observed for day 28; and
- λ_{2day} and ψ_{2day} are the actual failure and success rates observed for days 28 and 29.

We demonstrate a comparison between the traditional trust frameworks [11], [55] and Dirichlet distribution based trust (DDMT) [38], [39] with Protrust using resources τ_{evu}. The traditional trust framework estimates the probability that a volunteer host is able to successfully execute a task as $\tau(t) = \frac{\alpha+1}{\alpha+\beta+2}$, where α and β are the number of successful and failed interactions, respectively. The trust in DDMT is estimated as $\alpha_i / \sum_{j=1}^{k} \alpha_j$. We compare the ProTrust and traditional methods with the actual failure and success rates observed using the correlation coefficient *r* and standard deviation σ exploited in [62]–[64]. To determine the robustness and relativity of the individual model's trust τ_{evu}, we utilized the correlation coefficient for all the selected hosts during the first 27 days and the failure rate λ_i for the last two days of the trace usage data, defined as follows:

$$r = \frac{\Sigma(\tau_{evu} - \bar{\tau_{evu}})(\lambda_i - \bar{\lambda_i})}{\sqrt{\Sigma(\tau_{evu} - \bar{\tau_{evu}})^2 \Sigma(\lambda_i - \bar{\lambda_i})^2}}, \tag{26}$$

where τ_{evu} is a model's trust and λ_i is the failure rate. We also use the correlation coefficient *r* for the success rate ψ_i and compare this value with the estimated trust, as follows:

$$r = \frac{\Sigma(\tau_{evu} - \bar{\tau_{evu}})(\psi_i - \bar{\psi_i})}{\sqrt{\Sigma(\tau_{evu} - \bar{\tau_{evu}})^2 \Sigma(\psi_i - \bar{\psi_i})^2}}. \tag{27}$$

A model should have a positive or negative correlation between the trust value in the first 27 days and the failure and success rates in the last two days. If the trust of a volunteer host during the first 27 days is high, the failure and success rates for that volunteer host in the last 2 days are expected to be low, and vice versa. Additionally, a negative correlation is expected between the later failure rate and earlier trust. In other words, the correlation coefficient should be negative in the failure rate case and positive in the success rate case.

The correlation coefficients for the trust values of the compared frameworks during the first 27 days and the failure rates during the last 2 days are shown in Table 2. We discuss the correlation coefficients for all failure rate test cases (i.e., λ_{2day}, λ_{1day}, and λ_{5hour}) between the framework's trust value and the failure rate during the last 2 days compared to the traditional and DDMT frameworks. For example, in the case λ_{2day}, the correlation values for ProTrust and the other frameworks are 0.09, 0.04, and 0.01. Thus, all frameworks have positive values, which was unexpected. For the next day failure rate λ_{1day}, ProTrust's value is -0.15, which is more negative than the traditional framework's value of -0.09, whereas DDMT has a positive value. In the short prediction validation of the failure rate for the next 5 hours λ_{5hour}, ProTrust has a substantially more negative value (-0.33) than the two other frameworks because ProTrust relies on recent behavior and the $beta$ parameter considers more failure outcomes. In other words, the Beta distribution density curve shifts to accommodate the new outcomes, which is an appropriate mechanism to capture the behavior of a volunteer host who has recently become more malicious.

TABLE 2: Correlation Coefficients of Failure Rate

Framework	λ_{2day}	λ_{1day}	λ_{5hour}
ProTrust	0.09	-0.15	-0.33
Traditional	0.04	-0.09	-0.056
DDTM	0.01	0.013	-0.05

The correlation coefficients between the trust values of the compared framework and the success rates in the last 2 days are shown in Table 3. ProTrust's correlation coefficient for the case ψ_{2day} is 0.2084, which is larger than the traditional

models' correlation coefficient of 0.0983. DDMT shows the worst accuracy in the case of ψ_{2day}. For the success rates of the next day and next 5 hours ψ_{1day} and ψ_{5hour}, the trust value estimated by ProTrust is positive, while the trust value provided by the traditional framework is negative. In other words, as the success rate increases, the trust value provided by ProTrust increases more than the trust value provided by the other two frameworks.

Therefore, ProTrust is superior to the other frameworks, and the introduced concept of *trustworthiness affected by behavioral change* has a significant impact on the trust computation and updating of the Beta function parameters on recent outcomes.

TABLE 3: Correlation Coefficients of Success Rate

Framework	ψ_{2day}	ψ_{1day}	ψ_{5hour}
ProTrust	0.2084	0.1156	0.1440
Traditional	0.0983	-0.0065	-0.0548
DDTM	-0.05	0.06	-0.01

Next, we measure the amount of variation between the trust estimation and the actual failure to success ratio in terms of the standard deviation, as follows:

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(\tau_{evu} - \overline{\tau_{evu}})^2}. \tag{28}$$

In Table 4, the standard deviations for the failure rates of the ProTrust, traditional, and DDMT methods are 0.10, 0.07, and 0.06, respectively. For the long prediction case $\lambda_2 day$, ProTrust is much closer to the actual failure rate than are the other two frameworks. However, ProTrust shows the worst performance in the medium and short ($\lambda_1 day$ and $\lambda_5 hour$) predictions. However, according to the standard deviation of the success rate shown in 5, ProTrust is significantly better than the other two frameworks for $\lambda_1 day$ and $\lambda_2 day$ and much closer to the SD values for the success rate.

TABLE 4: Standard Deviation (SD) of the Failure Rate

ProTrust	Traditional	DDMT	λ_{2day}	λ_{1day}	λ_{5hour}
0.10	0.07	0.06	0.10	0.03	0.04

TABLE 5: Standard Deviation (SD) of the Success Rate

ProTrust	Traditional	DDMT	ψ_{2day}	ψ_{1day}	ψ_{5hour}
0.10	0.07	0.06	0.10	0.14	0.19

To validate ProTrust with the introduced concept of loyalty (e.g., different priorities of the task), we categorize the success rate of ψ into a number of categories:

- $\psi_{5hour}^{highPriority}$ as the success rate of high-priority tasks in the next 5 hours after day 27
- $\psi_{5hour}^{LowPriority}$ as the success rate of low- and medium-priority tasks in the next 5 hours after day 27
- $\psi_{1day}^{highPriority}$ as the success rate of high-priority tasks in the next 1 day after day 27
- $\psi_{1day}^{LowPriority}$ as the success rate of low- and medium-priority tasks in the next 1 day after day 27
- $\psi_{2day}^{highPriority}$ as the success rate of high-priority tasks in the next 2 days after day 27
- $\psi_{2day}^{LowPriority}$ as the success rate of low- and medium-priority tasks in the next 2 days after day 27

TABLE 6: Correlation Coefficients of the Compared Frameworks in terms of the Success Rate

Success Type	ProTrust	Traditional	DDMT
$\psi_{5hour}^{highPriority}$	0.1	-0.5	0.06
$\psi_{5hour}^{LowPriority}$	0.26	-0.05	-0.06
$\psi_{1day}^{highPriority}$	0.0	0.0	0.01
$\psi_{1day}^{LowPriority}$	0.22	-0.04	-0.02
$\psi_{2day}^{highPriority}$	0.0	0.0	0.0
$\psi_{2day}^{LowPriority}$	0.20	0.04	0.08

The correlation coefficients between the trust values of the compared frameworks in the first 27 days and different types of success rate in the last 2 days are shown in Table 6. As shown in Table 6, the correlation coefficients of ProTrust are positive and significantly different from those of the other two frameworks for long prediction $\psi_{2day}^{LowPriority}$. However, all the frameworks perform similarly in high-priority tasks for long prediction. The correlation coefficients of ProTrust are always better than those of the other two frameworks for medium- and low-priority tasks because of the adaptability of ProTrust to task priority.

Next, we measure the amount of variation between the trust estimation and the actual different types of success ratios in terms of the standard deviation. The standard deviations of the success rates shown in 7 indicate that ProTrust has better SD values in the long prediction case.

TABLE 7: Standard Deviation (SD) of the Success Rate

ProTrust	Traditional	DDMT	$\psi_{5hour}^{highPriority}$	$\psi_{5hour}^{LowPriority}$
0.10	0.07	0.06	0.19	0.21
ProTrust	Traditional	DDMT	$\psi_{1day}^{highPriority}$	$\psi_{1day}^{LowPriority}$
0.10	0.07	0.06	0.17	0.19
ProTrust	Traditional	DDMT	$\psi_{2day}^{highPriority}$	$\psi_{2day}^{LowPriority}$
0.10	0.07	0.06	0.13	0.14

Now, we categorize the failure rate of λ into various categories:

- $\lambda_{5hour}^{highPriority}$ as the failure rate of high-priority tasks in the next 5 hours after day 27
- $\lambda_{5hour}^{LowPriority}$ as the failure rate of low- and medium-priority tasks in the next 5 hours after day 27
- $\lambda_{1day}^{highPriority}$ as the failure rate of high-priority tasks in the next 1 day after day 27
- $\lambda_{1day}^{LowPriority}$ as the failure rate of low- and medium-priority tasks in the next 1 day after day 27
- $\lambda_{2day}^{highPriority}$ as the failure rate of high-priority tasks in the next 2 days after day 27
- $\lambda_{2day}^{LowPriority}$ as the failure rate of low- and medium-priority tasks in the next 2 days after day 27

TABLE 8: Correlation Coefficients of the Compared Frameworks

Success Type	ProTrust	Traditional	DDMT
$\lambda_{5hour}^{highPriority}$	-0.07	-0.07	-0.29
$\lambda_{5hour}^{LowPriority}$	-0.33	-0.06	0
$\lambda_{1day}^{highPriority}$	-0.08	-0.05	-0.03
$\lambda_{1day}^{LowPriority}$	-0.24	-0.07	0.05
$\lambda_{2day}^{highPriority}$	-0.05	-0.05	-0.01
$\lambda_{2day}^{LowPriority}$	-0.20	-0.03	0.02

The correlation coefficients between the trust values of the compared frameworks in the first 27 days and different types of failure rate in the last 2 days are shown in Table 8. As shown in Table 8, the correlation coefficients of ProTrust are negative and more strongly correlated than those of the other two frameworks for all cases of low- and medium-priority tasks. In the case of 5 hour prediction, DDMT performs well in high-priority tasks.

To conclude, the validation in most cases of short prediction proves that ProTust is a good solution for sensitive applications that require trusted volunteer hosts because the change detection method allows ProTrust to rely on recent behavior in the modeling phase. Moreover, the results show that ProTrust always performs better than the other frameworks and produces more accurate results for medium- and low-priority tasks because ProTrust incorporates different task priorities. Task heterogeneity, the priorities of actual and past tasks running on volunteer hosts, and the detection of behavioral changes of volunteer hosts make the ProTrust framework superior to the existing frameworks. In the next section, we discuss the change point computation used in the behavioral change detection method.

C. CHANGE POINT COMPUTATION

Figure 3 presents the behavior change of different volunteer hosts using horizontal lines for the fitted (underlying) mean. The black and white areas in each figure represent different behavior that occurs in the volunteer host (e.g., changes in behavior). Additionally, there are several horizontal red lines in each black and white area. Some volunteer hosts have a number of behavior changes. For example, Figure 3a has 6 behavior change points, while Figure 3b has 7 change points. The x- and y-axes represent time and data that are the outcomes for volunteer hosts, respectively. Figures 3a- 3f present the change behavior of different volunteer hosts. Each volunteer host has different behavior.

However, we are interested in only substantial changes, and in most cases, this change occurs at the last change point. If we observe change behavior in a volunteer host, ProTrust relies on only the recent behavior to estimate the trust. We use a portion of the cluster usage trace data to model ProTrust and the traditional framework. Thus, in the next section, we discuss the memory consumption of ProTrust compared to that of the traditional framework.

D. MEMORY CONSUMPTION

In this section, we discuss the memory consumption for ProTrust and the traditional framework. We assume the trust estimation is performed in the management server; therefore, the management server saves all volunteer host task outcomes in a *behavior* table. ProTrust and the traditional framework evaluate the trust based on past task outcomes (e.g., success and failure). Table 9 is the behavior table for the ProTrust framework, and Table 10 is the behavior table for the traditional framework.

TABLE 9: Behavior Table for ProTrust

Timestamp	Host ID	Past Interactions					
		t_{high}^{succ}	t_{med}^{succ}	t_{low}^{succ}	t_{high}^{unsucc}	t_{med}^{unsucc}	t_{low}^{unsucc}
4 bytes	8 bytes	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes

TABLE 10: Behavior Table for the Traditional Framework

Timestamp	Host id	Past Interaction	
		α	β
4 bytes	8 bytes	4 bytes	4 bytes

The behavior table for the traditional framework consists of four parameters, *timestamp*, which is the sampling time, Host identity as *Host ID*, and α and β, which are the numbers of successful and failed tasks. The data size for each host is $(m * 20)$ bytes, where m is the number of records. The memory requirement for the traditional framework for all volunteer hosts in bytes is $p(m * 20)$, where p represents the number of volunteer hosts.

For ProTrust, the behavior table contains 8 parameters, *timestamp*, which is the sampling time, Host identity as *Host ID*, where t_{high}^{succ} t_{med}^{succ} and t_{low}^{succ} are the total numbers of successful high-, medium-, and low-priority tasks and t_{high}^{unsucc} t_{med}^{unsucc} and t_{low}^{unsucc} represent the total numbers of unsuccessful high-, medium-, and low-priority tasks. The data size for each host is $(q*36)$ bytes, where q is the number of records: $q <= (m/1.8)$ because behavior changes result in the removal of some past task outcomes. Therefore, the memory requirement for ProTrust for all volunteer hosts is $p(q * 36)$ bytes, where p represents the number of volunteer hosts. The size of the behavior table depends upon the number of volunteer hosts and the length of past task outcomes.

Although the behavior table for the traditional framework consists of fewer parameters than that of ProTrust, ProTrust reduces the required memory by more than 65%. The results in Figure 4a are for 100 volunteer hosts. This graph shows that ProTrust consumes less memory than the traditional framework. For instance, for 100 volunteer hosts, the traditional framework consumes 11,391,000 bytes while ProTrust consumes 3,578,000 bytes. The data sizes appear to be very small; however, these data are for only 100 volunteer hosts and 27 days. This result shows that ProTrust is suitable for large-scale infrastructure.

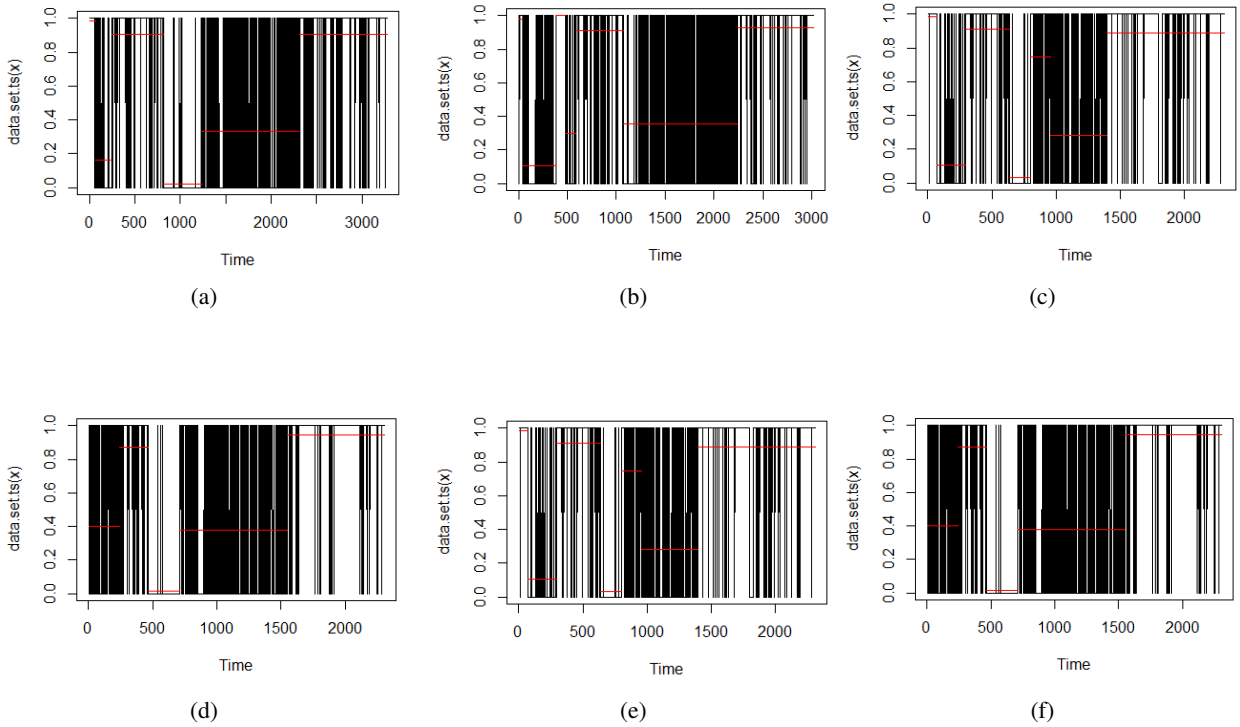


FIGURE 3: The volunteer hosts' data, with horizontal lines for the fitted (underlying) mean.

The results in Figure 4b illustrate the processing time, including the CPU time and elapsed time, to calculate the trust score in the ProTrust and Traditional frameworks. ProTrust takes less time than the traditional framework. For example, for 100 volunteer hosts, the CPU and elapsed processing times of the traditional framework are 78 and 110 ms, respectively, wheres for ProTrust, these values are 47 and 50 ms. Although the trust computation considers multiple parameters (e.g., the 8 parameters in Table 9), the size of data in the ProTrust computation is smaller than that in the traditional framework.

E. ADDITIONAL DISCUSSION

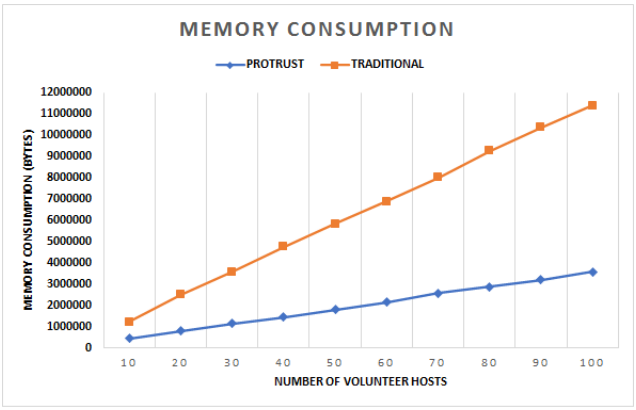
In this subsection, we briefly discuss some limitations of ProTrust and other related frameworks.

The quality of results produced by the volunteer host may affect the trust score. For example, two volunteer hosts who have completed the same task may have different results; one may take less time and produce a perfect result, while the other takes more time and produces a low-quality result. ProTrust cannot adequately capture this kind of information, as it relies on a binary result (e.g., failure and success). The quality of results is not the scope of this paper; however, it will be considered as future work to investigate the influence of quality of results on trust score calculation.

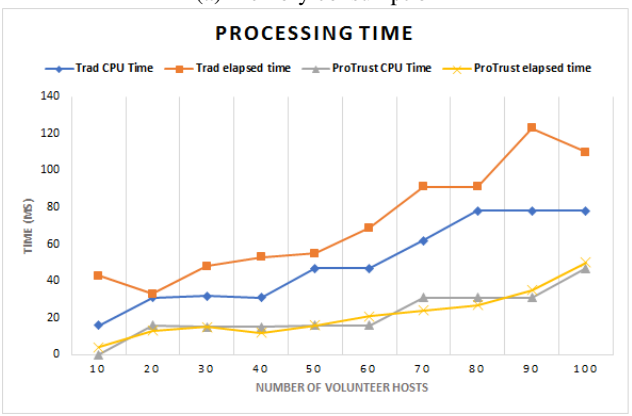
Moreover, in this paper, we assume that all the volunteer hosts function properly, and all the task failures result from untrusted volunteer hosts. VCC is inherently an open computing infrastructure, where the resources are donated by volunteers. This scenario translates into important technical challenges, such as the reliability of these resources. Simply finding the most trusted volunteer host does not guarantee that the volunteer host is reliable (e.g., hardware failure rate). For example, a trusted donor might have an incapable machine with a high failure rate or a malicious donor might have reliable hardware. Considering the reliability of volunteer host could improve the accuracy of the trust model.

IX. CONCLUSION AND FUTUREWORK

Volunteer cloud computing has been proposed as an economic way of undertaking tasks that are currently conducted by cloud computing models in specialized data centers. A number of crucial issues must be addressed for volunteer cloud systems, such as potential mistrust between volunteer hosts and user applications and fault tolerance. In this paper, we present ProTrust, a probability model for defining trust in a volunteer cloud environment. To model trust, we first present a new and improved version of the beta distribution. Next, we show how behavior can be observed to have different levels of loyalty. Then, we estimate trust with respect to the value that is expected based on past behavior. Finally, we present a behavior detection method to detect recent changes in behavior. To validate our framework, we utilize a large quantity of usage trace data from real-world applications



(a) Memory consumption



(b) Processing time

FIGURE 4: Memory consumption and Processing time

made available by Google. Our experimental results show that ProTrust produces precise results because it considers the impact of task priority and detects behavioral changes of volunteer hosts.

.

As future work, we are planning to address the influences of reliability and quality of results on ProTrust score calculation and integrate them via scheduling components in the real system. Finally, we plan to conduct future work in a scaling infrastructure computing system such as GENI [65] to test the feasibility of the proposed method.

REFERENCES

[1] H. Liu, L. Cao, T. Pei, Q. Deng, and J. Zhu, "A fast algorithm for energy-saving offloading with reliability and latency requirements in multi-access edge computing," IEEE Access, vol. 8, pp. 151–161, 2019.

[2] M. Chiang and T. Zhang, "Fog and iot: An overview of research opportunities," IEEE Internet of Things Journal, vol. 3, no. 6, pp. 854–864, 2016.

[3] H. Kellerer, U. Pferschy, and D. Pisinger, "Multidimensional knapsack problems," in Knapsack problems. Springer, 2004, pp. 235–283.

[4] B. Varghese and R. Buyya, "Next generation cloud computing: New trends and research directions," Future Generation Computer Systems, vol. 79, pp. 849–861, 2018.

[5] T. Ghafarian and B. Javadi, "Cloud-aware data intensive workflow scheduling on volunteer computing systems," Future Generation Computer Systems, vol. 51, pp. 87–97, 2015.

[6] T. Mengistu, A. Alahmadi, A. Albuali, Y. Alsenani, and D. Che, "A" no data center" solution to cloud computing," in 2017 IEEE 10th International Conference on Cloud Computing (CLOUD). IEEE, 2017, pp. 714–717.

[7] Z. Noorian and M. Ulieru, "The state of the art in trust and reputation systems: a framework for comparison," Journal of theoretical and applied electronic commerce research, vol. 5, no. 2, pp. 97–117, 2010.

[8] H. S. Narman, M. S. Hossain, M. Atiquzzaman, and H. Shen, "Scheduling internet of things applications in cloud computing," Annals of Telecommunications, vol. 72, no. 1-2, pp. 79–93, 2017.

[9] A. Rezgui, N. Davis, Z. Malik, B. Medjahed, and H. Soliman, "Cloudfinder: A system for processing big data workloads on volunteered federated clouds," IEEE Transactions on Big Data, 2017.

[10] A. Jøsang, R. Ismail, and C. Boyd, "A survey of trust and reputation systems for online service provision," Decision support systems, vol. 43, no. 2, pp. 618–644, 2007.

[11] A. Celestini, A. L. Lafuente, P. Mayer, S. Sebastio, and F. Tiezzi, "Reputation-based cooperation in the clouds," in IFIP International Conference on Trust Management. Springer, 2014, pp. 213–220.

[12] W. Feng, Z. Yan, H. Zhang, K. Zeng, Y. Xiao, and Y. T. Hou, "A survey on security, privacy, and trust in mobile crowdsourcing," IEEE Internet of Things Journal, vol. 5, no. 4, pp. 2971–2992, 2018.

[13] F. Hendrikx, K. Bubendorfer, and R. Chard, "Reputation systems: A survey and taxonomy," Journal of Parallel and Distributed Computing, vol. 75, pp. 184–197, 2015.

[14] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The eigentrust algorithm for reputation management in p2p networks," in Proceedings of the 12th international conference on World Wide Web. ACM, 2003, pp. 640–651.

[15] J. D. Sonnek and J. B. Weissman, "A quantitative comparison of reputation systems in the grid," in Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing. IEEE Computer Society, 2005, pp. 242–249.

[16] M. Eirinaki, M. D. Louta, and I. Varlamis, "A trust-aware system for personalized user recommendations in social networks," IEEE transactions on systems, man, and cybernetics: systems, vol. 44, no. 4, pp. 409–421, 2014.

[17] Q. Xu, Z. Su, S. Yu, and Y. Wang, "Trust based incentive scheme to allocate big data tasks with mobile social cloud," IEEE Transactions on Big Data, 2017.

[18] P. Zhou, S. Jiang, A. Irissappane, J. Zhang, J. Zhou, and J. C. M. Teo, "Toward energy-efficient trust system through watchdog optimization for wsns," IEEE transactions on Information Forensics and Security, vol. 10, no. 3, pp. 613–625, 2015.

[19] G. V. Crosby and N. Pissinou, "Cluster-based reputation and trust for wireless sensor networks," in Consumer Communications and Networking Conference, 2007. CCNC 2007. 4th IEEE. IEEE, 2007, pp. 604–608.

[20] G. V. Crosby, L. Hester, and N. Pissinou, "Location-aware, trust-based detection and isolation of compromised nodes in wireless sensor networks." IJ Network Security, vol. 12, no. 2, pp. 107–117, 2011.

[21] U. Ahmed, I. Raza, and S. A. Hussain, "Trust evaluation in cross-cloud federation: Survey and requirement analysis," ACM Computing Surveys (CSUR), vol. 52, no. 1, pp. 1–37, 2019.

[22] R. K. Ko, P. Jagadpramana, M. Mowbray, S. Pearson, M. Kirchberg, Q. Liang, and B. S. Lee, "Trustcloud: A framework for accountability and trust in cloud computing," in 2011 IEEE World Congress on Services. IEEE, 2011, pp. 584–588.

[23] S. Rizvi, K. Karpinski, B. Kelly, and T. Walker, "Utilizing third party auditing to manage trust in the cloud," Procedia Computer Science, vol. 61, pp. 191–197, 2015.

[24] S. M. Habib, V. Varadharajan, and M. Mühlhäuser, "A trust-aware framework for evaluating security controls of service providers in cloud marketplaces," in 2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications. IEEE, 2013, pp. 459–468.

[25] K. Hwang and D. Li, "Trusted cloud computing with secure resources and data coloring," IEEE Internet Computing, vol. 14, no. 5, pp. 14–22, 2010.

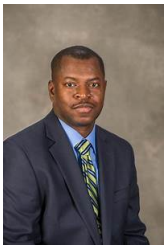
[26] R. Chen, J. Guo, and F. Bao, "Trust management for soa-based iot and its application to service composition," IEEE Transactions on Services Computing, vol. 9, no. 3, pp. 482–495, 2016.

[27] K. Zhao, S. Tang, B. Zhao, and Y. Wu, "Dynamic and privacy-preserving reputation management for blockchain-based mobile crowdsensing," IEEE Access, vol. 7, pp. 74 694–74 710, 2019.

[28] H. Kim, H. Lee, W. Kim, and Y. Kim, "A trust evaluation model for qos guarantee in cloud systems," International Journal of Grid and Distributed Computing, vol. 3, no. 1, pp. 1–10, 2010.

[29] V. C. Emeakaroha, K. Fatema, L. van der Werff, P. Healy, T. Lynn, and J. P. Morrison, "A trust label system for communicating trust in cloud services," IEEE Transactions on Services Computing, vol. 10, no. 5, pp. 689–700, 2016.
[30] X. Wu, R. Zhang, B. Zeng, and S. Zhou, "A trust evaluation model for cloud computing," Procedia Computer Science, vol. 17, pp. 1170–1177, 2013.
[31] X. Li and J. Du, "Adaptive and attribute-based trust model for service-level agreement guarantee in cloud computing," IET Information Security, vol. 7, no. 1, pp. 39–50, 2013.
[32] C. Mao, R. Lin, C. Xu, and Q. He, "Towards a trust prediction framework for cloud services based on pso-driven neural network," IEEE Access, vol. 5, pp. 2187–2199, 2017.
[33] X. Li, H. Ma, F. Zhou, and W. Yao, "T-broker: A trust-aware service brokering scheme for multiple cloud collaborative services," IEEE Transactions on Information Forensics and Security, vol. 10, no. 7, pp. 1402–1415, 2015.
[34] X. Li, H. Ma, F. Zhou, and X. Gui, "Service operator-aware trust scheme for resource matchmaking across multiple clouds," IEEE transactions on parallel and distributed systems, vol. 26, no. 5, pp. 1419–1429, 2014.
[35] N. Ghosh, S. K. Ghosh, and S. K. Das, "Selcsp: A framework to facilitate selection of cloud service providers," IEEE transactions on cloud computing, vol. 3, no. 1, pp. 66–79, 2014.
[36] L. Mashayekhy, M. M. Nejad, and D. Grosu, "A trust-aware mechanism for cloud federation formation," IEEE Transactions on Cloud Computing, 2019.
[37] C. J. Fung, J. Zhang, I. Aib, and R. Boutaba, "Dirichlet-based trust management for effective collaborative intrusion detection networks," IEEE Transactions on Network and Service Management, vol. 8, no. 2, pp. 79–91, 2011.
[38] W. Fang, W. Zhang, L. Shan, X. Ji, and G. Jia, "Ddtms: Dirichlet-distribution-based trust management scheme in internet of things," Electronics, vol. 8, no. 7, p. 744, 2019.
[39] B. Li, R. Lu, W. Wang, and K.-K. R. Choo, "Ddoa: A dirichlet-based detection scheme for opportunistic attacks in smart grid cyber-physical system," IEEE Transactions on Information Forensics and Security, vol. 11, no. 11, pp. 2415–2425, 2016.
[40] W. Ahmad, S. Wang, A. Ullah, Z. Mahmood et al., "Reputation-aware trust and privacy-preservation for mobile cloud computing," IEEE Access, vol. 6, pp. 46 363–46 381, 2018.
[41] Z. Lin and L. Dong, "Clarifying trust in social internet of things," IEEE Transactions on Knowledge and Data Engineering, vol. 30, no. 2, pp. 234–248, 2017.
[42] M. Wang, G. Wang, J. Tian, H. Zhang, and Y. Zhang, "An accurate and multi-faceted reputation scheme for cloud computing," Procedia Computer Science, vol. 34, pp. 466–473, 2014.
[43] Q. Wu, X. Zhang, M. Zhang, Y. Lou, R. Zheng, and W. Wei, "Reputation revision method for selecting cloud services based on prior knowledge and a market mechanism," The Scientific World Journal, vol. 2014, 2014.
[44] Z. Raghebi and M. R. Hashemi, "A new trust evaluation method based on reliability of customer feedback for cloud computing," in Information Security and Cryptology (ISCISC), 2013 10th International ISC Conference on. IEEE, 2013, pp. 1–6.
[45] Z. Tian, X. Gao, S. Su, and J. Qiu, "Vcash: A novel reputation framework for identifying denial of traffic service in internet of connected vehicles," IEEE Internet of Things Journal, vol. 7, no. 5, pp. 3901–3909, 2019.
[46] F. Moyano, C. Fernandez-Gago, I. Agudo, and J. Lopez, "A task ordering approach for automatic trust establishment," in International Symposium on Engineering Secure Software and Systems. Springer, 2012, pp. 75–88.
[47] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer, "Seti@ home: an experiment in public-resource computing," Communications of the ACM, vol. 45, no. 11, pp. 56–61, 2002.
[48] D. P. Anderson, "Boinc: A system for public-resource computing and storage," in Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on. IEEE, 2004, pp. 4–10.
[49] K. Watanabe and M. Fukushi, "Generalized spot-checking for reliable volunteer computing," IEICE TRANSACTIONS on Information and Systems, vol. 93, no. 12, pp. 3164–3172, 2010.
[50] K. Watanabe, M. Fukushi, and M. Kameyama, "Adaptive group-based job scheduling for high performance and reliable volunteer computing," Information and Media Technologies, vol. 6, no. 2, pp. 362–374, 2011.
[51] J. Cappos, I. Beschastnikh, A. Krishnamurthy, and T. Anderson, "Seattle: a platform for educational cloud computing," Acm sigcse bulletin, vol. 41, no. 1, pp. 111–115, 2009.
[52] J. Qiu, Z. Tian, C. Du, Q. Zuo, S. Su, and B. Fang, "A survey on access control in the age of internet of things," IEEE Internet of Things Journal, 2020.
[53] W. L. Teacy, J. Patel, N. R. Jennings, and M. Luck, "Travos: Trust and reputation in the context of inaccurate information sources," Autonomous Agents and Multi-Agent Systems, vol. 12, no. 2, pp. 183–198, 2006.
[54] W. Teacy, J. Patel, N. R. Jennings, and M. Luck, "Coping with inaccurate reputation sources: Experimental analysis of a probabilistic trust model," in Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems. ACM, 2005, pp. 997–1004.
[55] A. Josang and R. Ismail, "The beta reputation system," in Proceedings of the 15th bled electronic commerce conference, vol. 5, 2002, pp. 2502–2511.
[56] Z. Tian, C. Luo, J. Qiu, X. Du, and M. Guizani, "A distributed deep learning system for web attack detection on edge devices," IEEE Transactions on Industrial Informatics, vol. 16, no. 3, pp. 1963–1971, 2019.
[57] M. Li, Y. Sun, H. Lu, S. Maharjan, and Z. Tian, "Deep reinforcement learning for partially observable data poisoning attack in crowdsensing systems," IEEE Internet of Things Journal, 2019.
[58] Z. Tian, W. Shi, Y. Wang, C. Zhu, X. Du, S. Su, Y. Sun, and N. Guizani, "Real-time lateral movement detection based on evidence reasoning network for edge computing environment," IEEE Transactions on Industrial Informatics, vol. 15, no. 7, pp. 4285–4294, 2019.
[59] R. Killick, P. Fearnhead, and I. A. Eckley, "Optimal detection of changepoints with a linear computational cost," Journal of the American Statistical Association, vol. 107, no. 500, pp. 1590–1598, 2012.
[60] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: format+ schema," Google Inc., White Paper, pp. 1–14, 2011.
[61] S. Sebastio and A. Scala, "A workload-based approach to partition the volunteer cloud," in 2015 IEEE Conference on Collaboration and Internet Computing (CIC). IEEE, 2015, pp. 210–218.
[62] Y. Alsenani, G. V. Crosby, and T. Velasco, "Sara: A stochastic model to estimate reliability of edge resources in volunteer cloud," in Accepted tothe IEEE International Conference on Edge Computing (EDGE). IEEE, 2018.
[63] Y. Alsenani, G. V. Crosby, T. Velasco, and A. Alahmadi, "Remot reputation and resource-based model to estimate the reliability of the host machines in volunteer cloud environment," in 2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud). IEEE, 2018, pp. 63–70.
[64] X. Wang, F. Glaser, S. Herbold, and J. Grabowski, "On the relatively small impact of deep dependencies on cloud application reliability," in Cloud Computing (CLOUD), 2017 IEEE 10th International Conference on. IEEE, 2017, pp. 528–535.
[65] Geni. [Online]. Available: https://www.geni.net

YOUSEF S. ALSENANI is a PhD candidate at Southern University Illinois, Carbondale, USA. His research interests include trusted computing, reliability, and volunteer computing.



GARTH V. CROSBY is an associate professor at Texas A&M University. His research interests include wireless sensor networks, wireless body area networks, cyber (network) security, cyber-physical systems security and trust modeling.

KHALED R. AHMED is an Assistant Professor at Southern Illinois University, Carbondale. His research interests include high-performance computing, distributed and parallel computing, peer-to-peer computing, big data, machine learning, and image processing.

TOMAS VELASCO is an Associate Professor at Southern Illinois University, Carbondale. His research interests include reliability and applications of artificial intelligence to solve problems in industry.