Complex & Intelligent Systems (2022) 8:3747–3763
https://doi.org/10.1007/s40747-021-00518-3

ORIGINAL ARTICLE

A novel trust management model for edge computing

Rabia Latif[1] · Malik Uzair Ahmed[2] · Shahzaib Tahir[2] · Seemab Latif[3] · Waseem Iqbal[2] · Awais Ahmad[4]

Received: 31 March 2021 / Accepted: 26 August 2021 / Published online: 14 September 2021

Abstract
Edge computing is a distributed architecture that features decentralized processing of data near the source/devices, where data are being generated. These devices are known as Internet of Things (IoT) devices or edge devices. As we continue to rely on IoT devices, the amount of data generated by the IoT devices have increased significantly due to which it has become infeasible to transfer all the data over to the Cloud for processing. Since these devices contain insufficient storage and processing power, it gives rise to the edge computing paradigm. In edge computing data are processed by edge devices and only the required data are sent to the Cloud to increase robustness and decrease overall network overhead. IoT edge devices are inherently suffering from various security risks and attacks causing a lack of trust between devices. To reduce this malicious behavior, a lightweight trust management model is proposed that maintains the trust of a device and manages the service level trust along with quality of service (QoS). The model calculates the overall trust of the devices by using QoS parameters to evaluate the trust of devices through assigned weights. Trust management models using QoS parameters show improved results that can be helpful in identifying malicious edge nodes in edge computing networks and can be used for industrial purposes.

Keywords Edge computing · Edge devices · Cloud computing · Internet of Things · Trust management models

Introduction

With the emergence of technologies such as cloud computing, fog computing, and edge computing, the world has moved towards the centralization of data. Cloud service providers enable the remote storage of data centrally which could be accessed by all devices, thus removing the dependency of operating system (OS) and filesystem [3]. A single instance of a file can be accessed on a laptop, smartphone, and tablet, hence providing smart solutions to access the data. With the inception of IoT smart devices generating relatively large volumes of data, the Cloud can handle the data storage and processing needs, but the main bottleneck is the bandwidth of the network that carries data to the Cloud and back when required. Processes demanding real-time processing on the data require low response times. If each device sends real-time data to the Cloud for processing, the response time would increase exponentially. Therefore, to cater to such needs, a cloudlet or data center is required which can provide the IoT devices the ability to store and process data near to their locations. This technique is known as edge computing.

Edge computing is the paradigm shift in the cloud computing. It changes the approach of using the Cloud with IoT devices, ensuring real-time processing by providing a

✉ Shahzaib Tahir
shahzaib.tahir@mcs.edu.pk

Rabia Latif
rlatif@psu.edu.sa

Malik Uzair Ahmed
uzairmalik.msis14@students.mcs.edu.pk

Seemab Latif
seemab.latif@seecs.edu.pk

Waseem Iqbal
waseem.iqbal@mcs.edu.pk

Awais Ahmad
Aahmad.marwat@gmail.com

1 College of Computer and Information Sciences, Prince Sultan University, Riyadh 11586, Saudi Arabia

2 Department of Information Security, College of Signals, National University of Sciences and Technology (NUST), Islamabad, Pakistan

3 School of Electrical Engineering and Computer Science, National University of Sciences and Technology (NUST), Islamabad, Pakistan

4 Department of Computer Science, Air University, Islamabad, Pakistan

cloudlet/data center near the edge of the data source [2]. This layer accommodates devices that can perform analysis and data storage. The storage on these devices can either be permanent or temporary depending upon the nature of implementation [1]. This is where edge computing plays a vital role by minimizing problems such as low battery constraints, saving bandwidth, maximizing response time, privacy, and data safety, even though its roots go back to 1990 with the introduction of Content Delivery Networks (CDNs) [4]. Edge computing technology introduces an intermediate layer between the Cloud and the IoT devices.

A general view of edge computing architecture [5] shows that it comprises of four functional layers: edge devices, edge networks, edge servers, and core infrastructure. These layers perform the following functions:

1. *Edge devices (End users)*: Edge network includes numerous devices connected to the edge network which are data producers as well as consumers, e.g., IoT devices.
2. *Edge network*: Whole infrastructure including servers, devices, and core infrastructure is connected by internet, data center network, and wireless network.
3. *Edge servers*: Infrastructure providers own and provide edge servers which are responsible for delivering virtualized managing services. Edge data centers are also deployed which are connected to the traditional Cloud.
4. *Core network*: Network access such as internet, mobile network, management functions and computing services by centralized Cloud is provided by core network.

Along with multiple opportunities provided by edge computing, there are many challenges such as data security and privacy [5], trust, edge node computation, offloading and partitioning, service quality, deployment strategies, workload, and policies [6]. Trust is the final overall grade of the device, dependent upon the ratings provided by other devices. Trust can be viewed as opinion of the community regarding a device based on all their interactions with that device. The focus of this research will be to highlight trust management issues in Edge Computing architecture, study existing trust management systems developed for edge computing, and finally propose a trust management system and to evaluate existing schemes with the proposed scheme. As our reliance on IoT devices is increasing, the shift towards edge technology is inevitable [7]. In IoT, the trust of the smart devices could be maintained by the Cloud, whereas in edge computing, the smart devices intercommunicate data with each other to mitigate latency [8]. Therefore, a trust management mechanism is required to maintain the legitimacy of the device and the data provided by these devices, to detect data/information flow from rogue devices.

Contributions

Through this research, the following contributions are made:

– Highlighting of trust management issues in edge computing architecture, performing a comprehensive study of existing systems and models already proposed for solving issues regarding trust management.
– Proposing a novel trust management system to ensure the security of data and strong trust reliance on edge devices for fast and efficient communication and processing.
– Implementation and evaluation of the proposed trust management system and comparison of existing systems with the proposed model.

The remaining paper is organized as follows: "Literature review" gives a comparative analysis of the existing trust models. "Proposed trust management model" describes the architecture and working of the proposed framework and explains the proposed methodology. "Implementation and results" shows the results obtained after the implementation of our model. "Conclusion" concludes the paper and highlights future directions.

Literature review

Security risks such as replay attacks, message tampering, and forging are always there to discourage users from utilizing edge nodes for computing [9]. Hence, there is a great urge to maintain the trust of edge computing and many trust models are developed by scholars to tackle this problem.

Yuan and Li [11] introduced a trust model solely for serving edge computing that can be used for computing at a larger scale. Their multi-source feedback model adopted the idea of global trust degree (GTD) which is direct trust and feedback trust from brokers and edge nodes, by introducing three main layers: network, broker, and the device layer. Feedback was generated from edge devices as well as service brokers, hence named as multisource. Global convergence time (GCT) was used for the evaluation of efficiency. Experiments were conducted using NetLogo event simulator and personalized similarity measure (PSM). To compute reliability, task failure ratio (TFR) was computed.

Ruan et al. [10] proposed a trust model to evaluate applications as well as a node in a network which additionally in real-time help in resource configuration using measurement theory. Two metrics were defined: one to measure the quality of probability termed as trustworthiness and to evaluate trustworthiness with measure error, confidence was introduced. In the framework, multiple levels of trust were taken into consideration, such as trust of devices, tasks, and device-to-device trust. Apart from this, a new trust assessment algorithm was introduced to evaluate the model and a dynamic way to allocate resources for a task using a trust threshold value and avoiding redundancy. Further results and implementation were not presented. Alsenani et al. [12] demonstrated a model, SaRa (A Stochastic Model to Estimate Reliability of Edge Resources in Volunteer Cloud) targeting volunteer cloud computing and in this scenario CuCloud (Volunteer Computing as a Service (VCaaS) System), which was a client/server architecture including volunteer machines and dedicated servers. It was a probabilistic model that was based on the estimation of the reliability of nodes by exploiting the behavior of nodes. Main parameters included task behavior and characteristics, e.g., success, fail, priority, etc. Although the model had a random probability distribution, to validate this approach, Google clusters were used and the testing environment contained hundreds of machines. Compared to other probabilistic models [13], SaRa achieved greater precision.

There is a constant need that manufacturers of smart services provide configuration updates, control commands, and send and receive status information. In this case, Industrial IoT controllers need to protect themselves from unauthorized tampering and ensure the accuracy of inputs. To tackle such a problem, Pinto et al. [14] demonstrated a trust mechanism for edge devices in an industrial IoT environment to achieve confidentiality, integrity, and authenticity at both hardware and software levels. Since Trust Zone is gaining massive attention due to Advanced RISC Machine (ARM) processors, the usage of Trust zone-based architecture was a sensible choice that implemented a Trusted Execution Environment (TEE) in a slightly modified Real-Time Operating System (RTOS) but no further implementation was presented. Mobile phones connected to wireless networks comprise more than half of IoT devices and these devices are vulnerable to many security threats. Therefore, Rehiman and Veni [15] focused on a Model for data privacy and proposed a trust management framework for all three layers of IoT architecture which are application, network, and sensor layers. The architecture revolved around a security manager with enough memory and processing capacity to perform all tasks, minimizing overload for resource-constrained devices. Zero-knowledge protocol, access control mechanism, context-aware location privacy and Elliptic Curve Cryptosystem were some of the techniques used by the system for authentication along with public key generation and distribution by the security manager. Moreover, for packets anonymity and confidentiality, layer encryption scheme and data origin authentication scheme were proposed in the model. The model was simple and addressed all challenges, but no proper demonstration and evaluation was carried out. Furthermore, more than half of the computation was carried out by the security manager which, if it fails, brings down the whole system.

A trust management model based on centralized architecture for IoT was proposed by Alshehri and Hussain [16] that relied on a supernode that works like that of a router and additionally monitors the whole network in a clustered environment led by master nodes, supervising cluster nodes. Supernode consists of three modules: (a) Application Programming Interface (API) module that provides an interface between cluster nodes and master nodes for communication; (b) trust management module that allows trust communication between supernodes, master nodes, and cluster nodes by providing authentication data; and (c) trust communication module that allows two types of communication consisting of trust messages and value messages for establishing trust values. This model is unique for suggesting a centralized approach, but it also comes with its disadvantages. Alshehri and Hussain [28] proposed a trust management system based on Fuzzy security protocol. The trust metrics used takes direct and indirect trust scores and routing scores into consideration. Also, in [29], Alshehri et al. proposed a distributed trust management model inspired by the clustering technique consisting of Cluster Node component and the Master Node component.

Kim and Keum [17] proposed an IoT trust domain to protect IoT infrastructure from malicious attacks through a trustworthy gateway system. This system could be used for smart homes and smart offices. The system used a gateway system to pass the IP addresses, which were then converted to IDs. ID table was used as a repository to store ID information of all connected devices in the network. Theoretically, the system worked fine compared to an untrusted domain, but no implementation was provided.

Asiri and Miri [18] presented a model that used distributed neural networks to classify trustworthy nodes. In this model, they defined Alpha nodes that are more capable of controlling hubs, managing jobs, and do not frequently change. The functionality of nodes was considered to make clusters. The type of data being transmitted was considered at the profiling phase and used as one of the parameters for data security. Trustworthiness was determined based on the threshold rating provided by nodes. The main phases included data collection, virtual clustering, weight calculation, transaction, trust computation, node classification, and rate apprise. Though the system seems reliable, no general demonstration of the model was presented.

Mendoza and Kleinschmidt [19] offered a model that identified malicious nodes based on the services they chose to provide. At first, all nodes were assigned zero trust value and the process of neighbor discovery was started by sending announcement packets. When a node provided a service, its trust value increased and if the node was unable to do so, then its trust value decreased. This trust scheme was implemented in Cooja simulation provided by Contiki OS and detection of malicious nodes was successful. To share information, nodes share credentials for verification involving third-party intrusion. However, in a tactical environment such as search and rescue missions or military operations, access provision is limited. Also, reliability on hardware and early share of credentials is not possible. To solve these problems, Echevería et al. [20] proposed a model that uses key generation and distribution for disconnected environments using tactical cloudlets that allow data-staging, filtering, forward deploying, and data collection points. The proposed trust solution uses Identity-based Cryptography (IBC), Stanford Identity-based Encryption (IBE), and OpenSSL ciphers. To evaluate the system, a threat model by Microsoft Security Development Lifecycle (SDL) is used which proposes 60 potential threats out of which 14 are considered for the tactical environment. After implementation using open source tactical cloudlets 12 out of 14 threats were fully and partially handled.

Sharma et al. [21] proposed a generic trust management framework for IoT infrastructure. It defined all requirements to compute the trust of edge devices with update and maintenance. A unique concept of trustor and trustee was considered to evaluate the system. The whole system consisted of four phases: (a) the first phase gathered information through different parameters such as experience, reputation, and knowledge; (b) models were used for trust computation including machine learning, flow, fuzzy, probabilistic, and statistical models; (c) two architectures centralized and decentralized, which were used for trust dissemination; (d) the final phase included update and maintenance which occurred in an event-driven and time-driven scenario. Since it is a generic framework, no proper implementation was presented.

Wang et al. [22] defined trust mechanisms as self-organizing items that take an informed decision based on trust status considering three main elements which are service, decision making, and self-organizing. In a typical IoT infrastructure, three main networks and layers are used mainly, named as a sensor, core, and application. The model used formal semantics-based language and fuzzy set theory to form trust. Results achieved were consistent with an ideal situation and even though no demonstration was given for the working implementation of the model, it laid the foundation for future models for IoT layered architecture.

Kagal et al. [23] presented a trust management scheme that restricted the redelegation of task without following a delegation protocol and dealt with permissions in a distributed environment of supply chain management. For this purpose, CIIMPLEX EECOMS is chosen as an experimental environment and security agents were used for verification and authentication based on ID and verification certificates by Certification Authority (CA) which further were used as tickets for access to resources. Permission to delegate a task to or by the agent was also given by security agents and a log of delegations was maintained using Prolog. Delegations addresses are group, time-bound, action restricted, strictly re-delegatable, and re-delegatable delegations.

Even though the existing approaches have brought us new concepts to evaluate trust, but the support of these approaches is weak as the implementation of the proposed framework is weak and, in some cases, not present. Moreover, the Quality of Service (QoS) parameters used in the presented model proposes novelty along with implementation in real-time yielding better results.

A detailed analysis of the existing models is shown in Table 1.

Proposed trust management model

The architecture of the proposed trust management model is shown in Fig. 1. The proposed model consists of two main modules: a rating management module and a trust calculation module.

The rating module computes ratings based on QoS parameters and multicriteria decision analysis, which in turn produces a covariance matrix and calculates Singular Value Decomposition (SVD), and Principal Component Analysis (PCA) vectors in the component analysis module. Data are transferred to the prediction module which predicts the trust for the requested device. Edge computing technology has vast applications, due to its ability to provide various benefits such as low latency, Cloud offloading, and saving bandwidth; thus, it will be adopted globally by replacing the existing IoT infrastructure, where edge computing has its benefits. It also suffers from problems such as maintaining the reliability of data provided by the devices, and thus trust management becomes an important factor that provides some insight into the device, whose data are being received. In environments such as IoT and edge computing, the devices are unaware of each other's location, and intention. A device could be sending malicious and wrongful data to other devices or causing problems such as DoS in the network. To reduce the impact of such devices on the network, a lightweight trust management model is required which calculates the trust based on the ratings of the device. The proposed model calculates the trust based on the ratings provided by the other devices where each device maintains its rating table for the devices it is communicating with. Similarly, edge servers or data centers also maintain a rating table that stores ratings from all the devices. Trust is calculated based upon those ratings derived from the quality of service parameters. Devices exhibiting bad QoS parameters can be classified as malicious based on their network behavior, as they could be engaged in an active DoS attack [27].

An overview of the process is explained in a flowchart presented in Fig. 2. In the first phase, a connection is established

Table 1 Analysis of models

Reference	Mechanism of model	Functions	Limitations
[11]	Rating generated through feedback by multiple resources, mostly edge nodes	Protection against bad mouthing attacks, suitable for large-scale computations, increased reliability and speed	Although the multisource feedback approach is an integral part of the trust calculation, the likely risk triggered by bad mouthing attack brought by the feedback mechanism is more serious owing to the feature of the feedback mechanism itself. This results in a quick decline in the reliability of trust results
[10]	Trust framework based on measurement theory including confidence and trustworthiness	Measurement of trust for nodes and applications help in configuring resources and reduced redundancy	The author assumes a fixed threshold without giving an appropriate reasoning and what is the impact of changing the threshold value on the results. The result and analysis section is missing, making it difficult to check the efficiency and accuracy of the proposed approach
[12]	SaRa: a probability distribution model based on the behaviour of nodes	Considers behaviour, task characterization and better reliability	Using random probability distribution is tedious and time consuming, especially when creating larger samples
[14]	Uses trusted execution environments (TEEs) enabled by ARM TrustZone	Fulfilling input output real-time requirements and securing IoT edge devices	The authors primarily focus on securing the IoTs edge devices in real-time environment and use ARM TrustZone for trust calculation. The authors did not include the results in order to show the effectiveness of the proposed mechanism
[15]	Trust protocols based on a central security manager and public key distribution	Authentication, context aware location privacy, confidentiality, layer encryption, and data origin authentication	The proposed model only considers a small and simple IoT network for calculating the trust. The article lacks a proper demonstration and evaluation of the proposed approach. Additionally, more than half of the computation is carried by the security manager which if fails, brings down the whole system
[16]	Trust management framework based on centralized system and super node	Trust for multiple nodes in an IoT environment using super node as security manager	As the authors are working with the clustered network which results in various attacks on the network during trust calculation, e.g., on–off attack, bad-service provider attack and con-behavior attack
[17]	Trustworthy gateway system based on trust domain in IoT	Protects from malicious attacks	The proposed trust model handles spoofing and Denial of Service (DoS) attacks and is based on individual device security techniques and logical addressing. In reality, the intruder can inject false data and make a repudiation attack possible in the trust domain
[18]	Distributed probabilistic neural networks (PNNs)	Tackles cold start problem, considers the sensitivity of data, better availability, fast response time	It is strictly reliant on the specific environment for exchanging the trust parameters. The proposed approach introduces higher communication and processing overhead to build a trust model
[19]	Distributed trust scheme by observing services of local nodes	Prevention of selective attacks	Relies on a third party to share credentials for verification
[20]	Trust system based on secure key generation targeting decentralized environment	No dependence on central authentication, early distribution of credentials, any hardware security, and tackling threats in a tactical environment	The proposed scheme is based on cryptographic algorithms which results in high computation and processing

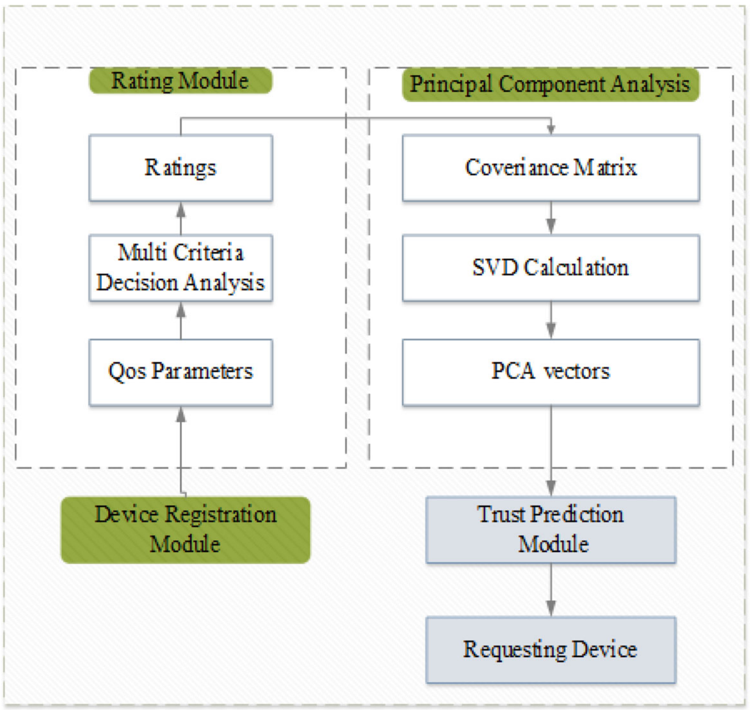


Fig. 1 Architecture of trust management model

and communication starts between edge devices. Rating is calculated based on these communications by computing the covariance matrix and single vector decomposition. The trust is predicted, and if the device is new, the cycle repeats at least five times to calculate the average trust for devices. A detailed description is presented below:

QoS parameters

In an edge computing model, each device can act as a server or a client. When a device is providing a service or data it is categorized as a server device, and in other scenarios where the device is gaining a service or data from another device, it is categorized as a client device. Based on this criterion, in our system, each client device at the end of communications would provide feedback on QoS parameters. The QoS parameters selected are utilized in classifying the network traffic pattern under DDoS attack in a real-time network [27]. Therefore, we can assume that devices that have bad QoS parameters are either faulty or malicious.

Ratings are derived from these QoS parameters using the multi-criteria decision analysis technique, and overall device trust is based upon these ratings. Trust in this model is an arithmetic value that lies in the range from 0 to 5, where 5 is an extremely trustworthy device and 0 is an extremely untrustworthy device.

The processes of our system start after every device has communicated at least once with each other. During the cold start, all devices connected to an edge server are registered by each edge server, and the information is forwarded to the Cloud. After registration, the system would be in observation mode where the observation mode would last for one transaction for each device.

Packet loss percentage

It is defined as the number of packets lost during the communication between two devices. High packet loss is considered bad for the network, and it would negatively effect the device rating, whereas low packet loss percentage is considered good for the network and positively effects the device rating. It can be calculated using (1) [20].

$$\text{PacketLossPercentage} = \frac{\sum_{i=0}^{n} \text{PL}}{\sum_{i=0}^{n} \text{PS}} * 100, \tag{1}$$

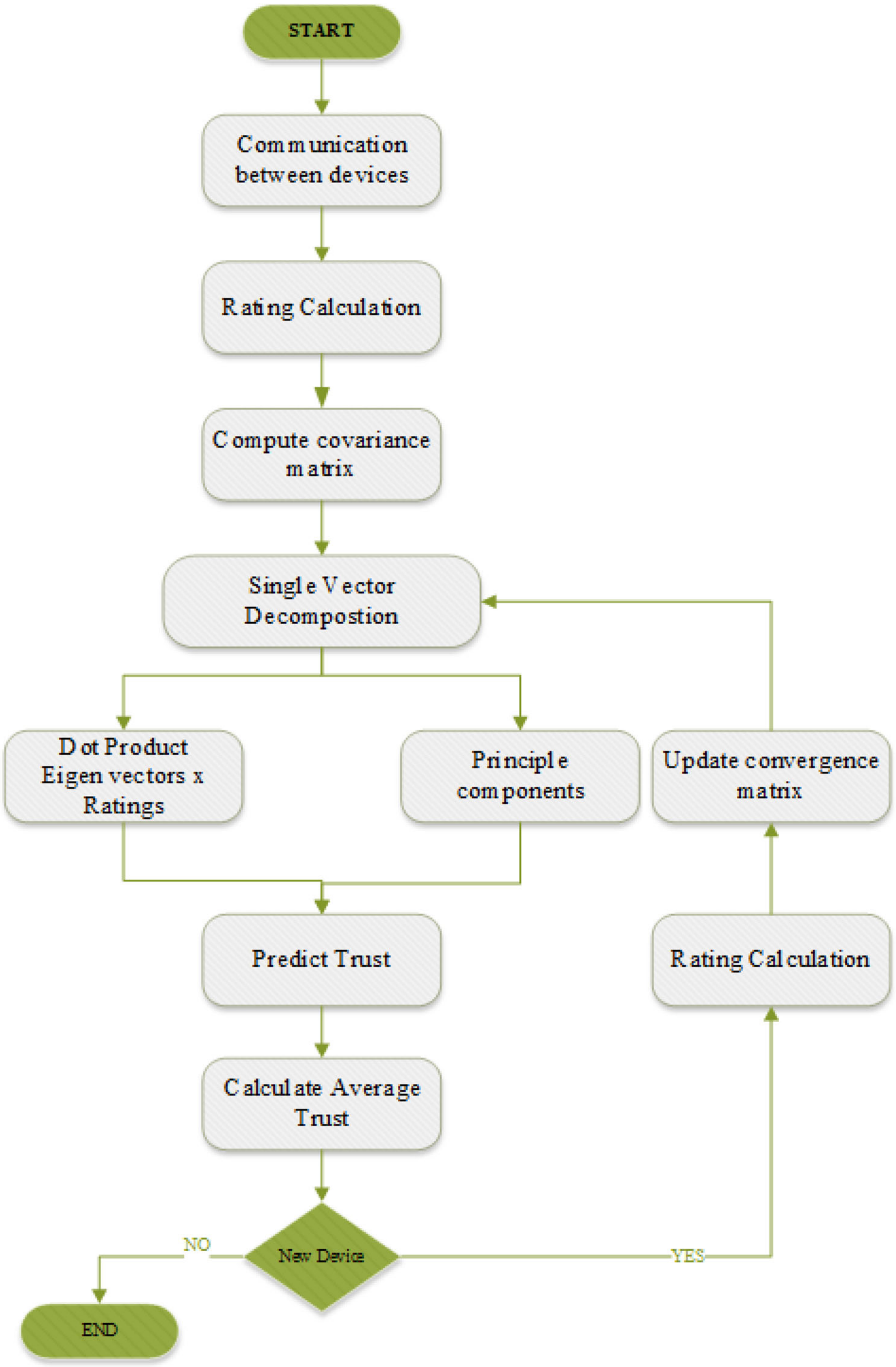
where PL = packets lost and PS = total packets sent.

Latency

Latency can be defined as the amount of time required for a packet to be transmitted from source to the destination. Latency depends on the congestion in the network. During the periods of high congestion, latency increases causing low rating. It can be calculated using (2) [20]

$$\text{Latency} = \sum (\text{PATime}_i - \text{PSTime}_i), \tag{2}$$

Fig. 2 Flow diagram of proposed system


START
Communication between devices
Rating Calculation
Compute covariance matrix
Single Vector Decompostion
Dot Product Eigen vectors x Ratings
Principle components
Update convergence matrix
Predict Trust
Rating Calculation
Calculate Average Trust
New Device
NO
YES
END

where PATime_i = packet arrival time and PSTime_i = packet send time.

Jitter (packet delay)

The variation in the time between the arrival of packets reaching the destination in a particular time frame. It indicates the consistency and stability of the network. It can be calculated by (3) [20]:

$$\text{Jitter} = \sum_{i=0}^{n} \left(\frac{\text{Delay}_i - \text{Delay}}{N} \right). \quad (3)$$

Table 2 Criteria for predicting ratings

No	Criteria
1	Latency
2	Packet loss percentage
3	Jitter
4	Throughput
5	Task failure rate

Throughput

Throughput is the number of bytes transferred from the source to destination. It is measured in bits per seconds unit (bps) using (4) [20]as follows:

$$\text{Throughput} = \frac{\sum_{i=0}^{n} (\text{Packets Received})}{\sum_{i=0}^{n} (\text{StartTime} - \text{StopTime})}. \tag{4}$$

Task failure ratio

Number of tasks that have been failed to be received by the client or generated by the server. This parameter is dependent upon the applications that are being run on the network. It can be calculated using (5):

$$\text{pft} = \left(\frac{\text{failed transactions}}{\text{total number of transactions}}\right) \times 100. \tag{5}$$

Multi-criteria decision analysis (MCDA)

The multi-criteria decision technique is used to decide on the best selection among various options based on the preferences of certain criteria. We explain MCDA in the scenario of our implemented system.

Defining criteria

Multiple quality of service parameters can be considered when communication is established between edge devices. Our experiment is based on the defined criteria as shown in Table 2.

Each parameter consists of some criteria that have a range of scores based on importance of resulting values [24].

We are considering five parameters including latency, packet loss percentage, jitter, throughput and task failure ratio. All these parameters have different units and they have separate criteria for contribution in calculation of ratings [30]. Hence, we define good and bad criteria for defining these parameters as depicted in Table 3. As shown in the table, those parameters which have an inverse effect on the performance of devices are already scored in inverse form; hence the beneficial criteria do not need to be divided with minimum value to make all parameters comparable. The measurement

Table 3 Points division for each rating criteria

Parameters	Criteria	Score
Latency	1. $<$ 30	5
	2. 30–50 ms	4
	3. 50–150 ms	2
	4. 150–200 ms	3
	5. $>$ 200 ms	1
Packet loss percentage	1. $<$ 4%	5
	2. 5–10%	4
	3. 10–15%	3
	4. 15–20%	2
	5. $>$ 20% and above	1
Jitter	1. $<$ 5 ms	5
	2. 5–10 ms	4
	3. 10–15 ms	3
	4. 15–20 ms	2
	5. $>$ 20 ms	1
Throughput	1. $>$ 90 Mbps	5
	2. 90–70 Mbps	4
	3. 70–50 Mbps	3
	4. 50–20 Mbps	2
	5. 20–0 Mbps	1
Task failure rate	1. $<$ 5 tasks	5
	2. 5–10 tasks	4
	3. 10–30 tasks	3
	4. 30–50 tasks	2
	5. $>$ 50 tasks	1

criteria presented in this table are based on research presented in [25].

When communication is established between devices, QoS parameters are recorded. We are calculating these parameters using edge computing simulation. As four parameters, i.e., jitter, packet loss percentage, task failure rate and Slatency are non-beneficial criteria in the contribution of ratings, their minimum values achieve highest score, whereas throughput is a beneficial parameter and hence its higher values get maximum score. The scoring is allocated from 1 to 5 because we have 5 parameters and want to get the final rating up to 5. For alternative scenarios where the user is considering more than 5 parameters, the scores can also be increased and vice versa.

After computing the score, we get values from 1 to 5 which are in the same unit; this keeps us from taking a range of values and normalizing it to get a weighted normalized decision matrix. Rating is denoted as follows:

$$R_i = \sum_{j=1}^{n} w_{ij} a_{ij}. \tag{6}$$

Table 4 Sample alternatives depicting rating criteria values

Devices	Throughput (Mbps)	Latency (ms)	Jitter (ms)	Packet loss percentage (%)	Task failure ratio
Device 1	1	280	35	35	55 tasks
Device 2	25	175	25	17	40 tasks
Device 3	55	75	15	12	20 tasks
Device 4	75	35	7	7	7 tasks
Device 5	95	25	2	3	3 tasks

Sum of weight of *j* number of parameters multiplied by *j* number of scores of parameters of device *i*, equals the rating of device *i*. The criteria weight is determined between 1 and 100% for each parameter based on its importance according to the scenario. The final rating obtained is used in the calculation of the average trust of a device.

Discussion on the proposed methodology

This section further elaborates our proposed scheme as an example scenario created from a subset of data, extracted from our main simulation, on the basis of best and worst case scenarios.

Calculating QoS parameters

Methodology for calculating QoS parameters is as follows: Determining Alternatives Values of QoS parameters are extracted from our main simulation as represented in Table 4. The values of each device are portrayed such that all the scenarios are covered ranging from best case scenario to worst case scenario of scores.
Assigning weight: We assign a relative weight to each parameter based on their importance in a given scenario. Weight of each QoS parameters can be assigned as per the requirement of the network. As for some networks, throughput of the devices would be much more important than other parameters and for others low task failure ratio would be more desirable. These values can be tuned according to the needs.

The sum of all weights must be equal to 1:

$$\sum_{i=1}^{n} w_i = 1. \tag{7}$$

We assign more weightage to those parameters which hold a strong position in the evaluation of trust among devices. The sum of weightage is always 100%. In this scenario, we have assigned an average weight, i.e. 0.20 to all parameters as shown in Table 5.
Value of scores: The parameters of each device are assigned a score based on its values recorded during the communication session described in Table 6.

Table 5 Weightage for rating calculation

Weightage	Parameters
0.20	Throughput
0.20	Latency
0.20	Jitter
0.20	Packet loss percentage
0.20	Task failure ratio

Final score: Multiply the weight assigned to each parameter with its score using (8) as shown in Table 7.

$$R_i = w_i a_i. \tag{8}$$

Final ratings: Final ratings for each device are obtained by sum of all final scores of QoS parameters of the device using (9) and (10) evaluated in Table 8:

$$R_{ij} = \sum_{j=1}^{n} w_{ij} a_{ij} \tag{9}$$

$$\begin{aligned} R_{ij} &= w_t\left(T_{ij}\right) + w_p\left(P_{ij}\right) + w_j\left(J_{ij}\right) \\ &\quad + w_t\left(L_{ij}\right) + w_f\left(F_{ij}\right). \end{aligned} \tag{10}$$

Table 9 shows the final ratings obtained for each device. It is observed that devices with lower scores have low ratings, whereas devices with higher score have high ratings.

Single vector decomposition

It is a matrix factorization technique, used mainly for dimensional reduction. It is used to reduce the dimensions of large data sets, while preserving as much information as possible. It can also be used in collaborative filtering [36]; collaborative filtering is a technique which predicts user preferences in a recommender system based upon the past user preferences [21]. In our system, we use collaborative filtering to find out trust of the device.

There are two main scores awarded to a device in the proposed scheme:

Table 6 Scores calculated based on sample alternatives

Devices	Throughput	Latency	Jitter	Packet loss percentage	Task failure ration
Device 1	1	1	1	1	1
Device 2	2	2	2	2	2
Device 3	3	3	3	3	3
Device 4	4	4	4	4	4
Device 5	5	5	5	5	5

Table 7 Multiplication of scores (Table 6) and weightage (Table 5)

Devices	Throughput	Latency	Jitter	Packet loss percentage	Task failure ration
Device 1	0.20 × 1	0.20 × 1	0.20 × 1	0.20 × 1	0.20 × 1
Device 2	0.20 × 2	0.20 × 2	0.20 × 2	0.20 × 2	0.20 × 2
Device 3	0.20 × 3	0.20 × 3	0.20 × 3	0.20 × 3	0.20 × 3
Device 4	0.20 × 4	0.20 × 4	0.20 × 4	0.20 × 4	0.20 × 4
Device 5	0.20 × 5	0.20 × 5	0.20 × 5	0.20 × 5	0.20 × 5

Table 8 Rating of devices

Devices	Throughput	Latency	Jitter	Packet loss percentage	Task failure ration
Device 1	0.10	0.10	0.10	0.10	0.10
Device 2	0.40	0.40	0.40	0.40	0.40
Device 3	0.60	0.60	0.60	0.60	0.60
Device 4	0.80	0.80	0.80	0.80	0.80
Device 5	1	1	1	1	1

Table 9 Final rating obtained by adding criteria for each device

Devices	Final ratings
Device 1	0.5
Device 2	2
Device 3	3
Device 4	4
Device 5	5

1. Ratings: The rating of a device is a value determined based on the communication between two devices. A device would be rated on the basis of its QoS parameters using formula 10. Ratings can be viewed as an opinion of one device based on its interaction.
2. Trust: Trust, on the other hand is the final overall grade of the device, dependent upon the ratings provided by other devices. Trust can be viewed as opinion of the community regarding a device based on all their interactions with that device.

The main difference between a rating and trust is that rating is computed by factors deriving from one-to-one communication between two devices. The rating would distinguish between the device being good or bad based on the analysis of one device. Even if many devices rate a single device, it would still lack the factor of input from the community.

In singular value decomposition, we take a rectangular matrix $X \times Y$ and decompose this matrix into three other matrices. Rating matrix serves as an input for SVD. In a rating matrix, all individual ratings of the device are mapped. Each device being rated is mapped in columns and rating of the respective device is mapped in rows. (Matrix is already given in Table 10)

$$A = USV^{\mathrm{T}}. \tag{11}$$

Since U is an $X \times Y$ orthogonal matrix so $U^T U = I_{n\times n}$. V is also an $X \times X$ orthogonal matrix hence $V^T V = I_{p\times p}$. Here I is the identity matrix. The diagonals of identity matrix are 1; all other values are 0.

Covariance matrix: Convergence matrix is calculated by combining the rating vectors. This step helps identify how variables are correlated. The convergence matrix is a symmetric matrix. To achieve this symmetry the following formula is utilized:

$$A = A \times \text{transpose}\,(A)$$

let:

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 1 & 0 \end{bmatrix} \Rightarrow AA^{\mathrm{T}} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$
$$= \begin{bmatrix} 1 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 1 \end{bmatrix} = B. \tag{12}$$

Compute eigenvalues of *A*

At the end of the process, the average trust of each device is calculated so that when a device which did not have a direct transaction with that device, it could determine the trust based on the past interactions of other devices. When a new device d_i enters the network, it is registered by the Cloud, and before it initiates communication with other device d_j, it could check the average trust value of that device, if it is higher than the threshold value 2.5. it is considered as a trusted device by the community or by the devices it has previously had transactions with. At the end of the communication, if d_i was a server device, it would be rated by the d_j. and these ratings would be forwarded to incremental Singular Vector Decomposition. Incremental SVD would find its trust based on the communications it had with other devices.

$$B\ x\ \lambda = \lambda\ x$$

so,

$$(B - \lambda I)x = 0. \tag{13}$$

Matrix reconstruction

The most significant eigen vectors are utilized to construct the final matrix. This matrix represents the final predicted trust values. The predicted values depend upon a criterion, i.e., the total number of the generated eigen vectors to be utilized, as the eigen vectors are arranged in descending order. Here, the first number shows highest significance as compared to the remaining values. During the matrix reconstruction the Dot Product ratings, Eigen values and ratings are calculated

$$A = \text{Dot}\,(\text{ratings, transpose}\,(U))\,. \tag{14}$$

Sort values by most significant number selected by some criteria. Values after certain threshold are discarded.

$$\text{PrT} = [A_{mxn}]\,[U_n]\,. \tag{15}$$

Incremental singular vector decomposition

The established network, up till now the network, has been setup with a fixed number of devices. One main feature to be tracked is, what happens when a new device k starts communicating with the edge nodes of our system. Of course, a whole new system cannot be established from scratch to observe the trust level for this new device k. Therefore, we have implemented a technique of incremental SVD for predicting the trust of the latest added devices to the network. This method is a continuity of SVD which we have presented previously, represented as Eq. (16).

We have the rating matrix R whose columns contain ratings of the devices.

Let

$$Z = \frac{U}{R} = U^{\mathrm{T}}R. \tag{16}$$

This is the orthogonal projection of R into U known as eigen coding.

Let

$$H = (I - UU^{\mathrm{T}})R = R - UZ. \tag{17}$$

This is the component of R which is orthogonal to the subspace spanned by R and I is the identity matrix.

Let

$$X = \frac{K}{H} = K^{\mathrm{T}}H. \tag{18}$$

In (19), K is an orthogonal basis of H and X is the projection of R onto the space orthogonal to U.

$$[U\ \ K]\begin{bmatrix} \text{diag}(s) & Z \\ 0 & X \end{bmatrix}\begin{bmatrix} V & 0 \\ 0 & I \end{bmatrix}$$
$$= \Big[U\ (I - UU^{\mathrm{T}})R/K\Big]\begin{bmatrix} \text{diag}(s) & U^{T}R \\ 0 & K \end{bmatrix}\begin{bmatrix} V & 0 \\ 0 & I \end{bmatrix}$$
$$= \Big[U'\ \text{diag}\ (s)V^{\mathrm{T}}C\Big] = [MR]. \tag{19}$$

As in single vector decomposition, the left and right matrices in the product are unitary and orthogonal. The middle matrix, denoted as D is a diagonal with a c-column border. We need to diagonalize D to update SVD.

$$U'\text{diag}(s)`V^{\mathrm{T}} \xleftarrow{\text{SVD}} D. \tag{20}$$

Devices with high score in QoS parameters have high average ratings, while devices with low QoS parameters have low average ratings.

Implementation and results

The proposed model is composed of two main modules, ratings module and trust calculation module. For ratings module, we are employing multi-criteria decision analysis approach and for the trust calculation module we are using singular vector decomposition. Trust is derived directly from ratings. For new devices on the network incremental SVD algorithm is employed.

The proposed system is implemented in MATLAB which supports matrix operations such as transpose and SVD. For simulating the communication between devices, we are using EdgeCloudSim, which is implemented in Java. EdgeCloudSim provides us with values which enable us to calculate our QoS parameters and derive our ratings by employing multi-criteria decision analysis technique. After extraction of the required parameters from EdgeCloudSim, they are stored in MySQL database which is accessed via a XAMPP server. After the ratings are calculated, they are then exported into MATLAB via CSV file for performing matrix operations.

Implementation

At first, we have computed trust for 10 devices and 100 after that. The need for an edge computing network to be trusted arises from the fact that it is a decentralized architecture, we assume that the Cloud is a trusted entity. Every device that is connected to the edge computing network is profiled. Our system works based on client and server, in edge computing a device can act as both a client and a server. Client devices have the power of server devices, so that after every service provided by the server device it is being rated based on its QoS parameters. These QoS parameters are translated into the ratings, ratings are being saved in our database. These ratings are then exported to MATLAB as a csv file, where we first calculate the transpose of the rating matrix R using Eq. (12).

This would result in a square matrix. This is a necessary requirement to determine the eigen vectors of a matrix. When SVD of a device is calculated, three parameters are gained from a single matrix. SVD is a method of decomposing a matrix into three other matrices represented in Eq. (11).

Example scenario

For the sake of example, we have taken, a subset of our main experiment, and included an example scenario where the communications between 10 devices are recorded.

Rating matrix

Rating matrix R was generated from the quality of service parameters as shown in Eq. (6). In our rating matrix, we can observe that some values are 0. This implies that devices have not communicated with each other.

This gives us a 9×10 rating matrix for 10 devices as shown in Table 10. To convert this into a symmetric matrix of 10×10; we multiply matrix R with R^{T}. This is because eigen values are generated only of a square matrix.

$$R(10 \times 10) = RR^{\mathrm{T}}. \tag{21}$$

Singular value decomposition

Singular value decomposition [26] technique is used to generate three other matrices from R using Eq. (11).

Since U is an $X \times Y$ orthogonal matrix so $U^{\mathrm{T}}U = I_{(n\times n)}$. V is also an $X \times X$ orthogonal matrix hence $V^{\mathrm{T}}V = I_{(p\times p)}$. Here I is the identity matrix. The diagonals of identity matrix are 1, all other values are 0.

$$RR^{\mathrm{T}} = USV^{\mathrm{T}}(USV^{\mathrm{T}}) = US^2V^{\mathrm{T}}RR^{\mathrm{T}}V = VS^2, \tag{22}$$

where V contains all eigenvectors and VS^2 contains all eigen values. Table 11 shows the experimental results from our implemented system.

S is a diagonal matrix which has entries only along the diagonal. It contains square roots of all eigenvalues of $R\ R^{\mathrm{T}}$ as shown in Table 12.

V is also an orthogonal matrix which contains eigen vectors of $R\ R^{\mathrm{T}}$ as described in Tables 13 and 14 and shows the predicted trust after matrix reconstruction.

Figure 3 presents the experimental results of trust calculation in our simulation system. These results were generated from the initial step by including ten devices for the experiment.

As it can be observed from the graph peaks that certain devices, i.e., device number 3, 5, 7 and 9, have comparatively higher trust value and other devices, i.e., device 1, 2, 4, 6, 8 and 10 have lower trust values. These results support our study explained that those devices which had high ratings based on quality of service parameters have turned out to be more trustworthy compared to low rated devices which had gained less score in the initial steps and have yielded lower values of trust (Table 15).

Experimental results

In our experiment, we have taken a network of 100 devices. All these devices communicate with each other in our simulation environment. Data from this simulation is extracted and QoS parameters are calculated, these parameters are gathered using multi-criteria decision analysis, bar chart of average ratings, average trust and scatter diagram of ratings and trust are shown as follows:

Average trust graph is shown in Fig. 4. The trust was computed according to the criteria previously presented in this research, the average ratings provide us a preliminary view

Table 10 Rating matrix after calculation of final ratings

3.8000	3.8000	4.6000	2.8000	3.2000	3.2000	2.6000	3.6000	3.4000
3.2000	3.8000	4.4000	0	3.4000	3.2000	3.8000	0	4.2000
2.8000	2.8000	4	3.4000	3.2000	4	4.4000	4.6000	0
0	3.2000	0	4	0	2.6000	0	3.2000	2.6000
3.4000	3.2000	4.8000	3.2000	2.8000	4.6000	3.2000	0	2.8000
4	0	3.2000	3.8000	2.6000	0	3.2000	4.4000	0
3.4000	3.8000	3.2000	2.8000	3.4000	2.6000	4.4000	2.8000	3
3.4000	3	0	3	0	4.2000	3	4.2000	4.2000
2.4000	3.6000	3	3.8000	3.6000	3.2000	3	0	0
4.8000	2.6000	4	0	3.8000	2.2000	3.2000	2.8000	4

Table 11 Resultant SVD experimental results *U*

−0.3761	0.0136	−0.0316	−0.1048	−0.4955	0.1549	0.2365	0.1721	0.6849	−0.1489
−0.3245	−0.4904	−0.2858	−0.0809	0.0624	0.1165	−0.1063	−0.4108	0.0950	0.5995
−0.3580	0.1914	0.3655	0.1666	0.3790	0.6387	0.3213	−0.0701	−0.1166	0.0233
−0.1755	0.5462	−0.3538	0.2101	−0.5147	0.1417	−0.0852	−0.1201	−0.3923	0.2023
−0.3477	−0.2401	−0.1270	0.3622	−0.0067	−0.4635	0.5376	−0.1694	−0.2422	−0.2878
−0.2568	0.2339	0.6702	−0.2549	−0.1569	−0.4543	−0.0460	−0.2073	−0.0320	0.2969
−0.3578	−0.0086	0.0027	−0.0467	0.0608	0.0556	−0.5957	−0.3904	0.0023	−0.5964
−0.2954	0.4737	−0.4145	−0.2388	0.5462	−0.2928	0.0253	0.1647	0.2021	0.0832
−0.2808	−0.1128	0.1431	0.6155	0.0599	−0.1269	−0.4187	0.5197	0.0772	0.1951
−0.3366	−0.2735	−0.0322	−0.5257	−0.1084	0.0905	−0.0190	0.5108	−0.4979	−0.0730

Table 12 Resultant SVD experimental results *S*

755.0756	0	0	0	0	0	0	0	0	0
0	58.3907	0	0	0	0	0	0	0	0
0	0	45.5991	0	0	0	0	0	0	0
0	0	0	31.1627	0	0	0	0	0	0
0	0	0	0	7.7302	0	0	0	0	0
0	0	0	0	0	6.2386	0	0	0	0
0	0	0	0	0	0	4.4993	0	0	0
0	0	0	0	0	0	0	1.5267	0	0
0	0	0	0	0	0	0	0	0.1370	0
0	0	0	0	0	0	0	0	0	3.8593e−16

Table 13 Resultant SVD experimental results *V*

−0.3761	0.0136	−0.0316	−0.1048	−0.4955	0.1549	0.2365	0.1721	0.6849	0.1489
−0.3245	−0.4904	−0.2858	−0.0809	0.0624	0.1165	−0.1063	−0.4108	0.0950	−0.5995
−0.3580	0.1914	0.3655	0.1666	0.3790	0.6387	0.3213	−0.0701	−0.1166	−0.0233
−0.1755	0.5462	−0.3538	0.2101	−0.5147	0.1417	−0.0852	−0.1201	−0.3923	−0.2023
−0.3477	−0.2401	−0.1270	0.3622	−0.0067	−0.4635	0.5376	−0.1694	−0.2422	0.2878
−0.2568	0.2339	0.6702	−0.2549	−0.1569	−0.4543	−0.0460	−0.2073	−0.0320	−0.2969
−0.3578	−0.0086	0.0027	−0.0467	0.0608	0.0556	−0.5957	−0.3904	0.0023	0.5964
−0.2954	0.4737	−0.4145	−0.2388	0.5462	−0.2928	0.0253	0.1647	0.2021	−0.0832
−0.2808	−0.1128	0.1431	0.6155	0.0599	−0.1269	−0.4187	0.5197	0.0772	−0.1951
−0.3366	−0.2735	−0.0322	−0.5257	−0.1084	0.0905	−0.0190	0.5108	−0.4979	0.0730

Table 14 Predicted trust after matrix reconstruction

3.2122	2.6677	3.1589	1.7190	3.7962	2.6446	3.3596	3.2522	3.5558	2.6830
2.0160	2.5342	1.3163	0.4822	1.1558	0.7529	2.4273	1.0730	1.5578	2.5186
4.4331	4.3594	4.0502	1.3545	4.0771	2.3451	3.6420	2.7187	2.1427	4.3438
2.8504	0.1519	3.2271	2.9440	2.5611	4.5275	2.8411	3.4638	2.8502	1.2943
3.6466	4.0521	2.2050	1.3041	3.2170	1.8638	3.7361	2.1069	2.6037	3.8932
2.4677	2.8875	2.9713	0.4523	4.0319	0.9050	1.8504	2.3360	2.2507	2.2302
3.3786	2.9625	3.2665	1.3782	2.8948	2.3025	3.1951	2.5278	2.3029	3.1867
3.4056	0.1091	4.9297	2.6249	0.5341	4.7246	2.8617	2.9581	0.8548	2.1846
3.8624	4.7804	1.9581	1.0932	3.9918	1.4450	3.6374	1.9020	2.4677	4.2231
1.5827	1.6542	2.1605	0.3975	1.4941	0.8523	1.7963	1.6443	1.7695	1.6525

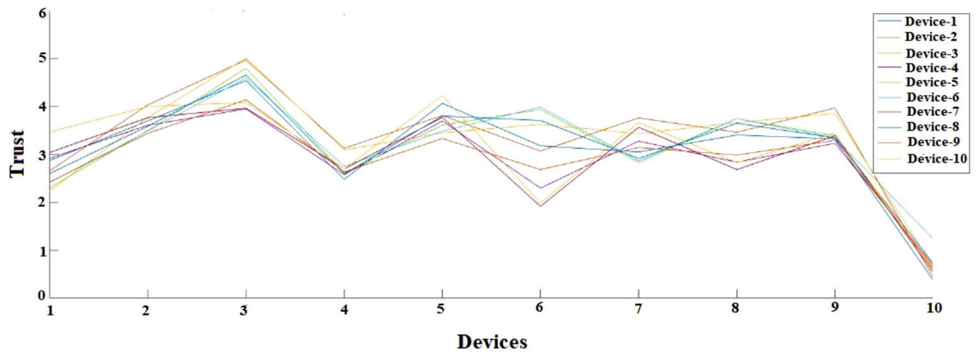


Fig. 3 Trust calculation graph

Table 15 Predicted trust P1 given by new device

3.5806	3.4299	2.4303	3.1555	3.3765	2.5852	3.6030	3.6116	2.4100	2.8575
3.7337	4.0960	3.7870	3.5393	3.6707	3.5181	3.7320	3.7283	3.3695	3.7245
3.6378	4.0658	4.7908	3.3912	3.8161	4.4587	3.0551	3.6551	3.9314	4.0839
3.2586	3.8867	3.2808	3.1601	3.3228	2.8452	3.5487	3.3503	2.9100	2.9944
3.5661	3.4004	3.3605	3.1192	3.4774	3.4925	3.0711	3.5525	2.9861	3.4095
3.7500	3.5686	3.7511	2.8896	4.2038	4.0112	2.6643	4.0732	3.0254	2.7139
3.2040	3.9245	3.4654	3.3326	3.0565	2.8952	3.6915	3.1272	3.1402	3.5468
3.6273	3.6471	3.7061	3.0738	3.8320	3.7523	2.9818	3.7819	3.1188	3.1366
3.1402	3.7833	3.8076	3.1357	3.1698	3.3101	3.2177	3.1260	3.2666	3.4851
3.7746	3.7696	1.4761	3.0619	3.9188	1.6816	4.1456	4.1932	1.6469	1.4127

of where device overall trust would fall. Thus, Fig. 4 provides us a general overview of the feedback given to a device by devices which it previously interacted with, in the form of ratings. Comparing both graphs in Figs. 4 and 6, we observe that devices have both positive correlation and negative correlation.

Figure 5 represents the ratings scatter graph for 1000 devices in a network. This scatter graph shows the position of the ratings for each device. In the sample space, according to the trends shown in this figure, most ratings lie in the middle, whereas the lowest ratings are near to 1.5 and a few devices have the highest rating of 5. Most of the ratings lie at an average distance from each other. Ratings show the estimation based on QoS parameters, but these ratings lack the factor of community input. Community factor is an important aspect when calculating trust rating. While comparing Figs. 4 and 6, a general overview of a trend can be observed, true picture and difference between ratings and trust can be observed by comparing the scatter graph shown in Figs. 5 and 7. The *x*-axis of the scatter graph show the devices, whereas the *y*-axis show ratings or trust in the case of Fig. 7. While the saturation in Fig. 5 is spread widely which represents the individual rating given to a device, whereas in Fig. 7 the saturation is near the mean position which reflects the community factor affecting the trust value given by each device.

Referring to the final predicted trust, our simulation in Fig. 6 shows that the average trust of each device is more than the mean value 2.5. If there is a device whose trust level falls below the mean level, such devices may be engaged in active DOS attacks. There are several approaches that could

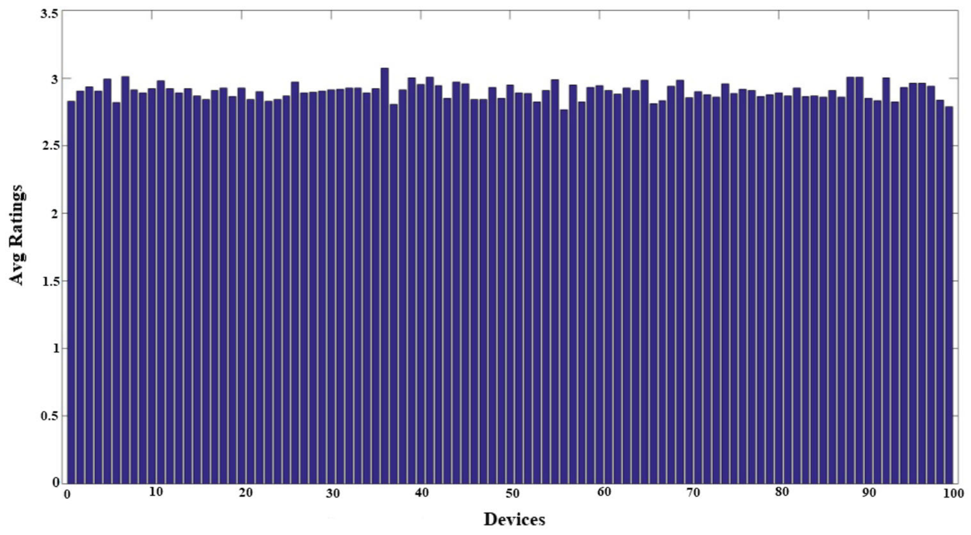


Fig. 4 Average rating graph

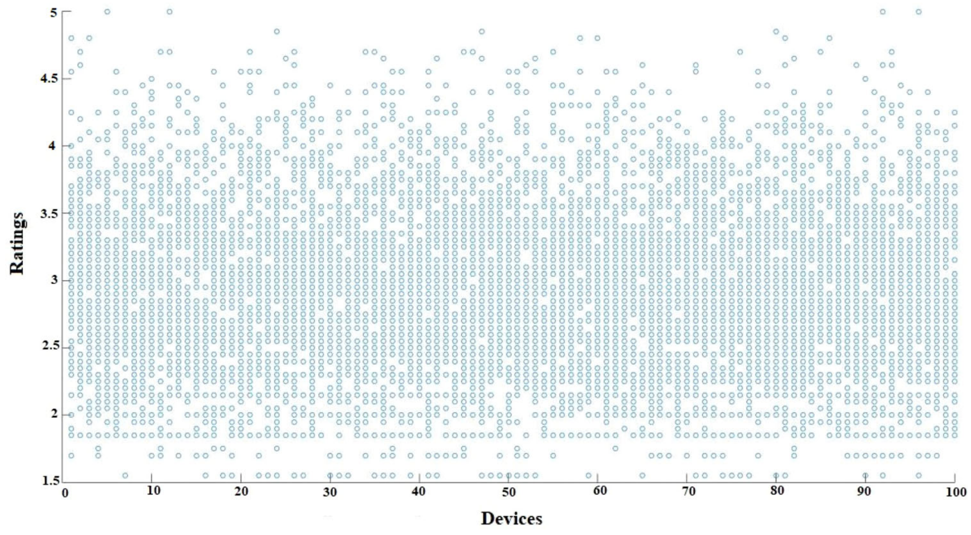


Fig. 5 Ratings scatter graph

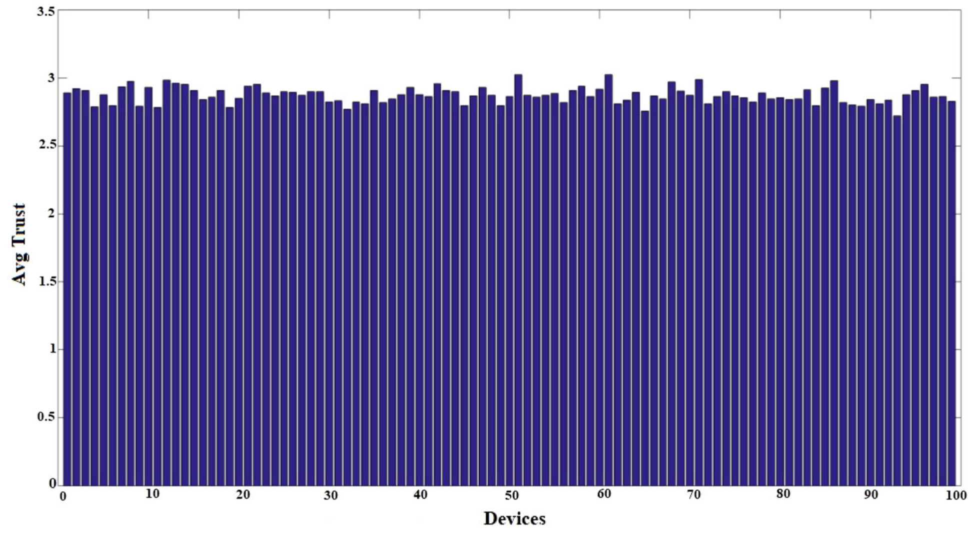


Fig. 6 Average trust graph

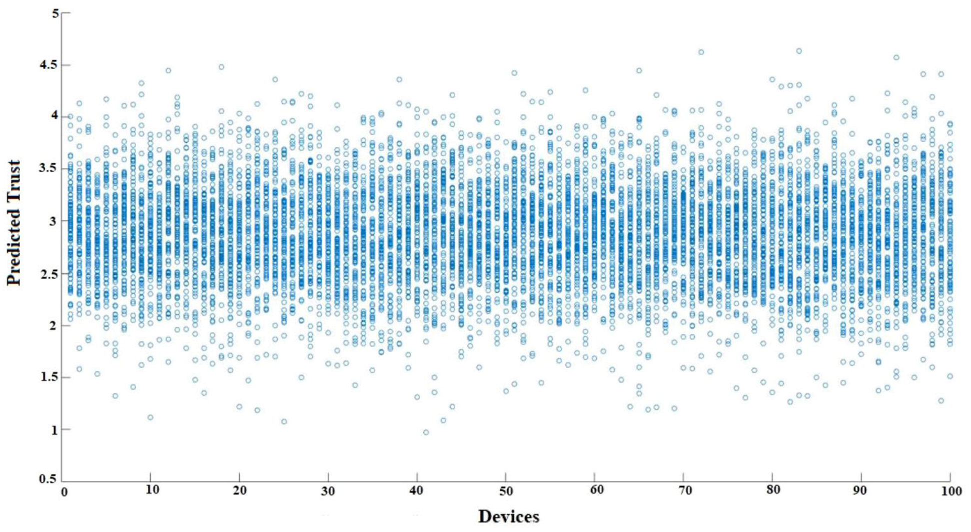


Fig. 7 Trust scatter graph

be utilized to reduce the impact of such devices on the network, and data.

- Such devices could either be allocated low network resources so their immediate impact may be removed from the network allowing other devices to continue communication smoothly.
- Such devices could be placed into isolation until the devices are checked for malfunctions, bad configuration, or patched for a vulnerability.

Conclusion

Edge computing as we know it today is an emerging technology where the generation, distribution, storage and computation of data is performed at the edge of the network. A big concern of bandwidth in cloud computing is also resolved using edge computing but new concerns such as privacy, security, latency, computation power at the edge and offloading need to be addressed. This research targets the most significant issue of the security and reliability of edge devices by proposing a trust management model to evaluate the credibility of edge nodes.The proposed model calculates the trust based on the ratings provided by the other devices. Each device maintains its rating table for the devices it is communicating with. Similarly, edge servers or data centers also maintain a rating table which stores ratings from all the devices. Trust is calculated based upon those ratings depending on the quality of service parameters such as packet loss percentage, latency, jitter, throughput and task failure ratio. Each parameter consists of some criteria that have a range of scores based on the importance of the resulting values and the weight is assigned to the devices accordingly. Trust management models using QoS parameters show improved results that can help identify malicious edge nodes in edge computing networks and can be used for industrial purposes.

Acknowledgements This work was supported by the Artificial Intelligence and Data Analytics (AIDA) Lab CCIS, Prince Sultan University, Riyadh, Saudi Arabia. Authors are thankful for the support.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Ren J, Pan, Goscinski A, Beyah RA (2018) Edge computing for the Internet of Things. IEEE Netw 32(1):6–7
2. Shi W, Cao J, Zhang Q, Li Y, Xu L (2016) Edge computing: vision and challenges. IEEE Internet Things J 3(5):637–646
3. Armbrust M, Armando F, Rean G, Anthony DJ, Randy K, Andy K, Gunho L, David P, Ariel, Ion S, Mate z. "A view of cloud computing." Communications of the ACM 53, no. 4 (2010): 50–58
4. Satyanarayanan M (2017) The emergence of edge computing. IEEE Comput Soc 50(1):30–39

5. Zhang J, Chen B, Zhao Y, Cheng X, Hu F (2018) Data security and privacy-preserving in edge computing paradigm: survey and open issues. IEEE Access 6:18209–18237
6. Varghese B, Wang N, Barbhuiya S, Kilpatrick P, Nikolopoulos DS (2016) Challenges and opportunities in edge computing. In: IEEE international conference on smart cloud (SmartCloud), pp 20–26. https://doi.org/10.1109/SmartCloud.2016.18
7. Yu W, Liang F, He X, Hatcher WG, Lu C, Lin J, Yang X (2017) A survey on the edge computing for the Internet of Things. IEEE Access 6:6900–6919
8. Garg SK, Versteeg S, Buyya R (2013) A framework for ranking of cloud computing services. Future Gener Comput Syst 29(4):1012–1023
9. Beck MT, Maier M (2014) Mobile edge computing: challenges for future virtual network embedding algorithms. In: The eighth international conference on advanced engineering computing and applications in sciences (ADVCOMP), pp 65–70
10. Ruan Y, Durresi A, Uslu S (2018) Trust assessment for Internet of Things in multi-access edge computing. In: IEEE 32nd international conference on advanced information networking and applications (AINA), pp 1155–1161. https://doi.org/10.1109/AINA.2018.00165
11. Yuan J, Li X (2018) A multi-source feedback based trust calculation mechanism for edge computing. In: IEEE conference on computer communications workshops (INFOCOM WKSHPS), pp 819–824. https://doi.org/10.1109/INFCOMW.2018.8406900
12. Alsenani Y, Crosby G, Velasco T (2018) Sara: a stochastic model to estimate reliability of edge resources in volunteer cloud. In: IEEE international conference on edge computing (EDGE), pp 121–124. https://doi.org/10.1109/EDGE.2018.00024
13. Jøsang A, Ismail R, Boyd C (2007) A survey of trust and reputation systems for online service provision. Decis Support Syst 43(2):618–644
14. Pinto S, Gomes T, Pereira J, Cabral J, Tavares A (2017) IIoTEED: an enhanced, trusted execution environment for industrial IoT edge devices. IEEE Internet Comput 21(1):40–47
15. Rehiman KR, Veni S (2017) A trust management model for sensor enabled mobile devices in IoT. In: International conference on I-SMAC (IoT in social, mobile, analytics and cloud)(I-SMAC), pp 807–810. https://doi.org/10.1109/I-SMAC.2017.8058290
16. Alshehri M.D., Hussain F.K. (2018) A Centralized Trust Management Mechanism for the Internet of Things (CTM-IoT). In: Barolli L., Xhafa F., Conesa J. (eds) Advances on Broad-Band Wireless Computing, Communication and Applications. BWCCA 2017. Lecture Notes on Data Engineering and Communications Technologies, vol 12. Springer, Cham.pp 533–543. https://doi.org/10.1007/978-3-319-69811-3_48
17. Kim E, Keum C (2017) Trustworthy gateway system providing IoT trust domain of smart home. In: IEEE ninth international conference on ubiquitous and future networks (ICUFN), pp 551–553. https://doi.org/10.1109/ICUFN.2017.7993848
18. Asiri S, Miri A, (2016) An IoT trust and reputation model based on recommender systems. In: 14th Annual conference on privacy, security and trust (PST), pp 561–568. https://doi.org/10.1109/PST.2016.7907017
19. Mendoza CV, Kleinschmidt JH (2016) Defense for selective attacks in the IoT with a distributed trust management scheme. In: IEEE international symposium on consumer electronics (ISCE), pp 53–54. https://doi.org/10.1109/ISCE.2016.7797367
20. Echeverría S, Klinedinst D, Williams K, Lewis GA (2016) Establishing trusted identities in disconnected edge environments. In: IEEE/ACM symposium on edge computing (SEC), pp 51–63. https://doi.org/10.1109/SEC.2016.27
21. Sharma A, Pilli ES, Mazumdar AP, Govil MC (2016) A framework to manage trust in internet of things. In: International conference on emerging trends in communication technologies (ETCT), pp 1–5. https://doi.org/10.1109/ETCT.2016.7882970
22. Wang JP, Bin S, Yu Y, Niu XX (2013) Distributed trust management mechanism for the internet of things. Trans Tech Publ Appl Mech Mater 347:2463–2467. https://doi.org/10.1109/ETCT.2016.7882970
23. Kagal L, Finin T, Cost RS, Peng Y (2001) A framework for distributed trust management. In: Second workshop on norms and institutions in multi-agent systems
24. Oktavianti E, Komala N, Nugrahani F (2019) Simple multi attribute rating technique (SMART) method on employee promotions. J Phys Conf Ser 1193:012028. https://doi.org/10.1088/1742-6596/1193/1/012028
25. Risawandi, Rahim R (2016) Study of the simple multi-attribute rating technique for decision support. Int J Sci Res Sci Technol 2:491–494
26. Sarwar B, Karypis G, Konstan J, Riedl J (2002) Incremental singular value decomposition algorithms for highly scalable recommender systems. In: Proceedings of the 5th international conference in computers and information technology
27. Mirkovic J, Reiher P, Fahmy S,Thomas R, Hussain A, Schwab S, Ko C (2006) Measuring denial of service, conference on computer and communications security. In: Proceedings of the 2nd ACM workshop on quality of protection, pp 53–58
28. Alshehri MD, Hussain FK (2019) A fuzzy security protocol for trust management in the internet of things (fuzzy-IoT). Computing 101:791–818. https://doi.org/10.1007/s00607-018-0685-7
29. Alshehri MD, Hussain F, Elkhodr M, Alsinglawi BS (2019) A distributed trust management model for the Internet of Things (DTM-IoT). In: Jan M, Khan F, Alam M (eds) Recent trends and advances in wireless and IoT-enabled networks. EAI/Springer innovations in communication and computing. Springer, Cham. https://doi.org/10.1007/978-3-319-99966-1
30. Phemius K, Bouet M (2013) Monitoring latency with OpenFlow. In: Proceedings of the 9th international conference on network and service management (CNSM 2013), 2013, pp 122–125. https://doi.org/10.1109/CNSM.2013.6727820