Trust Management Model for Edge Computing

by

Uzair Malik

A thesis submitted to the faculty of Information Security Department, Military College of Signals, National University of Sciences and Technology, Rawalpindi in partial fulfillment of the requirements for the degree of MS in Information Security

July 2019

THESIS ACCEPTANCE CERTIFICATE

Certified that final copy of MS Thesis written by **NS Uzair Malik**, Registration No. **00000117615**, of **Military College of Signals** has been vetted by undersigned, found complete in all respects as per NUST Statutes/Regulations/MS Policy, is free of plagiarism, errors, and mistakes and is accepted as partial fulfillment for award of MS degree. It is further certified that necessary amendments as pointed out by GEC members and local evaluators of the scholar have also been incorporated in the said thesis.

Signature: _____

Name of Supervisor **Asst Prof Dr. Rabia Latif**

Date: _____

Signature (HOD): _____

Date: _____

Signature (Dean/Principal) _____

Date: _____

Acknowledgments

I would like to thank Allah Almighty for bestowing me with countless blessings. I express my gratitude towards my reverent mentors, parents and faculty members of my prestigious institute. I would have not been able to accomplish this milestone without their inevitable support and guidance

I would also like to express special thanks to my supervisor; Assistant Professor Dr. Rabia Lateef who provided me tremendous guidance and supervision throughout my academic session and for her help throughout my thesis.

I would also like to pay special thanks to CTO of my company for his tremendous support and cooperation. I appreciate his patience and guidance throughout the whole thesis.

Finally, I am grateful to my institution for providing me a platform and resources to achieve this milestone and I would like to express my gratitude to all the individuals who have rendered valuable assistance to my study.

Uzair Malik
Aug 2019.

Abstract

Edge computing is a distributed architecture, that features decentralized processing of data near the source, where data is being generated, or by the device that is generating the said data. These devices are known as Internet of Things (IoT) devices or edge devices. The data generated by the IoT devices has increased significantly over the passage of time, it has become infeasible to transfer all the data over to the cloud for processing. This gave rise to the edge computing paradigm. In edge computing data is processed by these devices and only the required data is sent to 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 system is required that maintains the trust of a device and manages the service level trust along with the quality of service (QoS). This allows the communication of trusted devices and their data can also be considered as legitimate as the devices are trusted.

Since this is an emerging paradigm, a limited amount of research is carried out to address the trust issues. Therefore, the focus of this research is to develop a lightweight trust management model which calculates the overall trust of the devices by using QoS parameters to evaluate the trust of devices through assigned weight. 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.

Table of Contents

List of Figures

List of Tables

ACRONYMS

IoT	Internet of Things
WAMP	Web Application Messaging Protocol
REST	Representational State Transfer
P2P	Peer to Peer
CDN	Content Delivery Network
RSU	Road Side Units
VR	Virtual Reality
TEE	Trusted Executed Environment
GTD	Global Trust Degree
GCT	Global Convergence Time
PSM	Personalized Similarity Measure
TFR	Task Failure Ratio
API	Application Programming Interface
SVD	Single Value Decomposition
QoS	Quality of Service

C h a p t e r 1

INTRODUCTION

1.1 Introduction

With the advent of technologies such as cloud computing, the world has moved towards a centralization of data, cloud provided service of storing data, and logic at a singular centralized point which could be accessed by all the devices, thus removing the dependency of OS, and filesystem from the structure, a single instance of a file could be accessed on your laptop, smartphone and tablets, it provided smart solution to access all your data. With introduction of IoT smart devices, which generated a relatively large volume of data, cloud can provide the data storage and processing needs, but the main bottleneck is the bandwidth of the network that carry data to the cloud and back. Processes demanding real time processing on the data require low to minimum response times, if each device sends real-time data cloud for processing the response time would increase exponentially. Therefore, to cater for such needs, a cloudlet or datacenter was required which could provide the IoT devices ability to store and process data near their locations, this technique has been named as edge computing. Edge computing is the paradigm shift in the cloud computing architecture, it changes the approach of using the cloud with IOT devices, ensuring real-time processing by providing a cloudlet/data center near the edge of the data source.

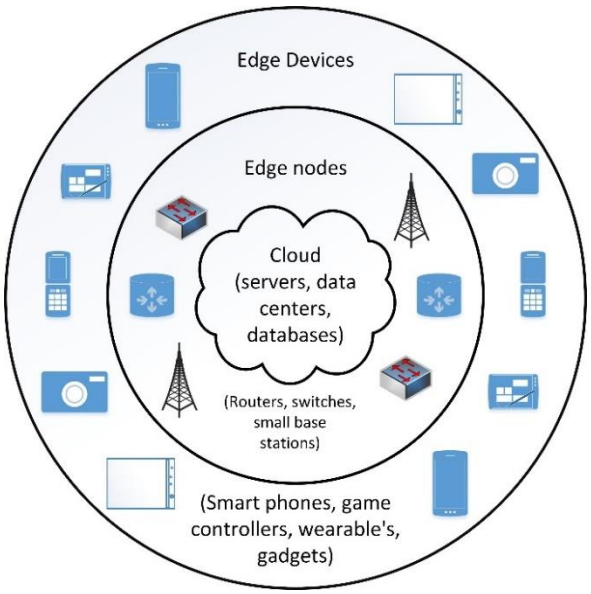

Figure 1.1.1: Relation of Cloud with Edge nodes and Edge devices

Edge computing technology introduces an intermediate layer between the cloud and the IoT devices, this layer accommodates devices which have capabilities to perform analysis and data storage, the storage on these devices can either be permanent or temporary depending upon the nature of implementation can be seen in Figure 1.1 [1].

A large amount of data is generated through IoT devices and cloud computing is not enough to deal with it resulting in the increase popularity of edge computing. Number of devices were recorded to be 15.41 billion in 2015 and is expected to get doubled at the end of 2020. This leads to several issues such as delay, jitter, congestion and many other privacy concerns such as data generated by home sensors or surveillance. In addition, cloud computing consumes many resources and network also suffers with congestion and long poor latency rate. To solve these problems edge computing has been introduced that includes devices at the edge for computation and analysis which are time dependent and improves latency rate [2].

Apart from the increase in devices, social media applications like Instagram, twitter, WhatsApp, Facebook and many more have also led to the generation of huge pile of unstructured data. So, what's a better way to handle this edge data at edge level by optimizing cloud techniques which can expand from a single home to a whole city where sensors act as an intermediary to control traffic. Cloud computing utilizes many services such as platform, infrastructure and software services which can also be used by edge devices to analyze the data and make smart choices in accordance with the analyzed data since the edge act as data center without taking orders from cloud. All this processing in real time that is processing, cashing, computing and delivery hence, reducing cloud traffic [3].

Edge computing platform always contain edge nodes and cloud side. Cloud side manages message routing and database. Edge side consists of monitoring, analysis and responsible for generating messages [4].

1. **Cloud Dashboard-** Request handler module manages three main parts of cloud dashboard and manages requests by users through maintaining a copy and queuing.

   These include:

- **User Interface**- This provides web applications to access remote application and perform functions like create, start, stop, remove or deploy. Dashboard plays the

part of interface for users to interact with applications. Users can login to dashboard and access services provided by edge nodes. Users can also deploy edge nodes through dashboard.

- **Messaging Service**- This provide security and reliability in addition to communication between nodes. It transmits the message to the destination node by accessing internet. Edge nodes are registered with the server. Web Application Messaging Protocol (WAMP) is used to establish messaging service and dynamic IP.
- **Database-** Results of processed information are stored in database. In addition, location and details of user are also stored. For modularity five components are used such as image, device, user, storage and compute. These modules are interlinked and use REST APIs allowing portability and platform independence.

2. **Edge Node-** Nodes at the edge of network are regarded as edge nodes that also perform several functions.

   These include:

- **Container Management-** It is a virtualization method for edge nodes and is considered light weighted. It consists of libraries, and program comprising of file system utilizing less RAM with low cost and supporting dense applications. It also allows users to run applications using a complete software package. Techniques such as swarm Kubernetes and fleet are used to deploy such systems. Nodes are monitored through these systems. After successful execution of applications, some program allow container to be deleted.
- **Messaging Service-** Web application Messaging Protocol (WAMP) is used mainly for communication of messages to and from server.

Edge centric computing is considered as a novel approach that will shift forward the working of services, data and computing applications from central approach to the edge of the network. It will most certainly retain advantages of cloud keeping the decision power to the edge devices giving more way to human centric applications.

Centralized approaches usually consist of cloud at the core and data centers. Content distribution networks, desktop PCs, smart phones, web servers and Nano data centers are also the part of the network along with the usage of sensor with IP addresses and embedded devices.

Edge computing comprises of many factors such as proximity, intelligence, trust, control and humans in the edge. Proximity in considered as a valid argument of P2P network system and CDNs. Since it is more feasible to deliver data to closed edge nodes than to far way servers in a logical and physical manner. Intelligence is always in spotlight whenever we talk about sensors, increased capacity and miniaturization. These factors pave way for crowdsensing and human controlled applications. Data security is always a big concern of every network. Hence, trust of edge devices should be controlled and handled by edge devices for better management. Coordination and management of devices in a synchronized manner is another part of edge computing where control is edge is of high importance. Similarly, humans are always present at the edge of the network. Therefore, giving human more power over their data and crowdsourcing of information lead to many innovative opportunities [5].

In general, there are three types of edge devices in edge computing which are:

1. **Local Devices**- deployed for set purpose, easy to deploy and manage.
2. **Localized Datacenters**- On site datacenters, generally closer to IOT devices, capable of processing and data storage.
3. **Regional Datacenters**- located relatively closer to IOT devices then the centralized cloud.

There are many advantages to the deployment of edge computing such as [6]:

- Faster response times
- Control over sensitive data
- Reliable operations with intermittent connectivity
- Interoperability (Interoperability between legacy and modern devices)

Drawbacks in edge computing are as follows [6][7]:

- System reliability
- Security and privacy

Edge computing is most commonly implemented in following two ways [8]:

1. **Full Cloud Storage** – In this approach all the data generated by the IoT's is stored in cloudlets or edge servers, is sent to the cloud.
2. **Partial Cloud Storage** – In this approach sensitive data is sanitized and only non-sanitized data is sent to the cloud for storage.

Both these approaches result in different implications for trust management in edge computing. In one case cloud acts as a centralized control point for entire infrastructure therefore a centralized trust management system can be deployed which manages trust values for each of the edge servers or cloudlets, where as in second approach cloud server acts only as a backup agent for non-critical information. Distributed trust management system is required to ensure data provided by each edge server can be trusted. Both these approaches have their merits and are employed as per organizational needs. A full cloud storage provides a backup of whole system in case an edge server goes down whereas the second approach provides better privacy and security of the data and reduces the dependence on cloud and mitigates vulnerability to the cloud outages.

1.2 Research Significance

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 proposed scheme.

1.3 Relevance to National Needs

As our reliance on IoT devices is increasing, the shift towards edge technology is inevitable. In IoT technology 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. Therefore, a trust management mechanism is required to maintain legitimacy of device and data provided by these devices, this way information from rouge devices can be detected.

1.4 Motivation and Problem Statement

Edge computing is a distributed architecture, that features decentralized processing of data near the source, where data is being generated, or by the device that is generating the said data. These devices are known as Internet of Things (IoT) devices or edge devices. The data generated by the IoT devices has increased significantly over the passage of time, it has become infeasible to transfer all the data over to the cloud for processing. Since these devices contain enough storage and processing power, it gave rise to the edge computing paradigm. In edge computing data is processed by these

devices and only the required data is sent to 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 system is required which maintains the trust of the device and manages the service level trust along with the quality of service. This allows the communication of trusted devices and their data can also be considered as legitimate as the devices are trusted.

Since this is an emerging paradigm a limited amount of research is carried out to address the trust issues. Therefore, the focus of this research is to develop a lightweight trust management model which calculates the overall trust of the device and service level trust management model to manage trust levels for each service provided by these devices. The proposed model will be evaluated and compared with existing models.

1.5 Contributions and Outcomes

This thesis made several contributions which are enlisted below with their brief description and shown in figure 1.2.

- **Contribution 1:** Highlighting of trust management issues in edge computing architecture.
- **Contribution 2:** Comprehensive study of existing systems and models already proposed for solving issues regarding trust management.
- **Contribution 3:** Proposing a trust management system to ensure security of data and strong trust reliance on edge devices for fast communication and processing.
- **Contribution 4:** Implementation and evaluation of proposed trust management system and comparison of existing systems with the proposed model.

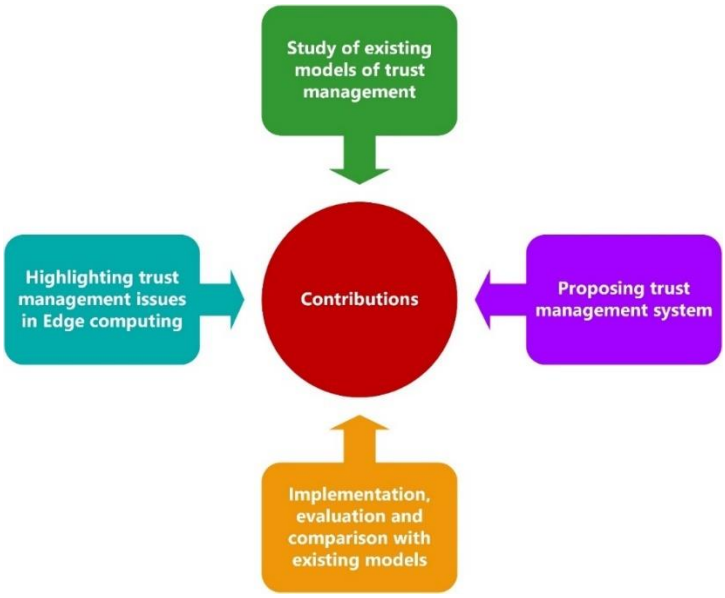

Figure 1.2: Thesis Contributions

1.6 Thesis Outline

The thesis is structured as follows:

- **Chapter 2** discusses the history, basic concepts, evolution, integration and adaptability of Edge Technology. Trust management models and their comparative analysis along with challenges, need for trust management and advantages have been presented.
- **Chapter 3** discusses the proposed framework for trust management system. The architecture, its individual modules, implementation technique along with key components are also presented.
- **Chapter 4** discusses the execution mechanism of the proposed trust management system and its effective results.
- **Chapter 5** will conclude the research. It will also highlight the future implications of proposed system along with its importance in industrial development.

1.7 Conclusion

The aim of this research is to perform a profound analysis of existing as well as proposed trust management systems in Edge computing which will soon consume existing IoT infrastructure of new era encouraging us to find better solutions. Research methodology, importance, and overall structural organization of thesis is also mentioned.

C h a p t e r 2

LITERATURE REVIEW

2.1 Introduction

With the advancement of Internet of Things (IoT) and prodigious use of cloud services, edge computing has been introduced where processing of data is possible at the edge of network.

We have been using IoT for supply chain management since 1999 and then it became a part of our lives by mingling with other fields such as transport, home, healthcare and environment [9]. All these new paradigms gave rise to the problem of storage, processing and retrieval of data globally where cloud computing has played a marvelous role as our savior since its rise around 2005 [10]. Now in the prime stage many challenges arise that require minimum burden on networks, quality of data, and short response time. This is where edge computing plays a vital role by minimizing problems such as low battery constraint, 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) [11].

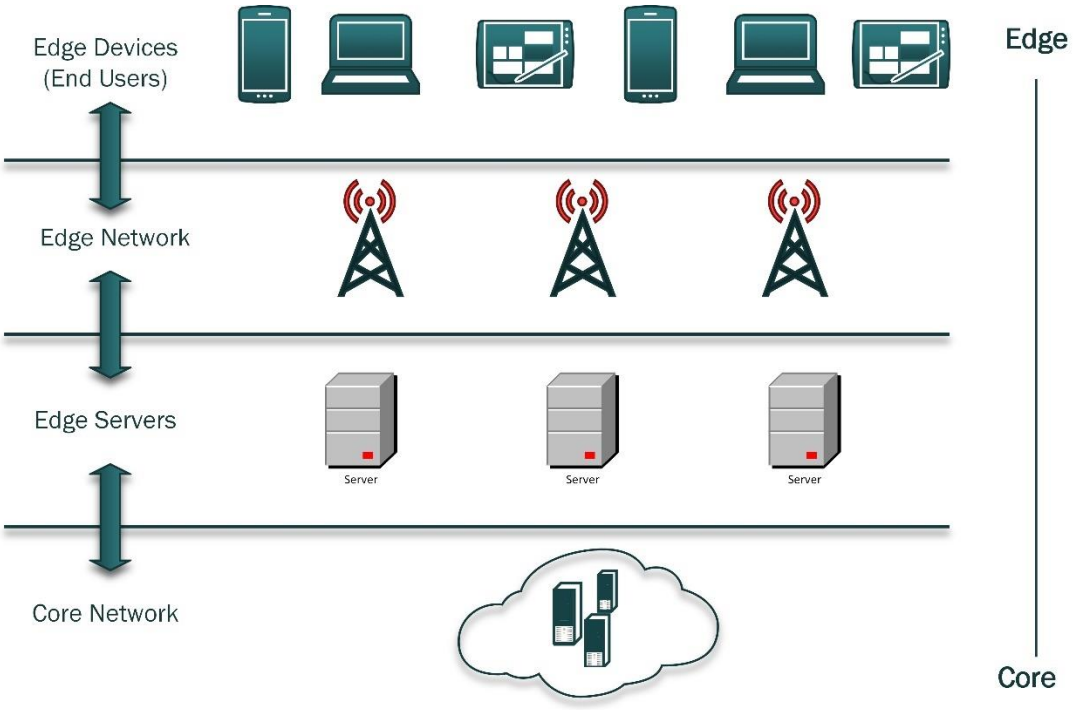

Figure 2.1: Edge Computing Architecture

Figure 2.1 shows us a general view of edge computing architecture [12], edge devices, edge networks, edge servers and core infrastructure make up four functional layers. These layers perform following functions:

- **Edge devices (End users)-** Edge network include numerous devices connected to edge network which are data producers as well as consumers e.g. IoT devices.
- **Edge network-** Whole infrastructure including servers, devices, core infrastructure is connected by internet, data center network and wireless network.
- **Edge servers-** Infrastructure providers owe and provide edge servers which are responsible for delivering virtualized managing services. Edge datacenters are also deployed which are connected to the traditional cloud.
- **Core network-** network access such as internet, mobile network, management functions, computing services by centralized cloud is provided by core network.

2.2 Why Edge Computing?

With edge computing we have minimized latency rate and bottleneck of bandwidth where cloud computing is involved. Apart from these, there are several reasons that require edge computing as an optimal solution.

2.2.1 Decentralized Cloud

Distribution of applications geographically has led us to choose decentralized approach as compared to centralized cloud computing where we can process data near the source and reduce latency of network [13]. Many web-based applications benefitting mobile users uses edge computing to improve the quality of service. Apart from this real time gamming is possible due to analytics being perform at one hop distance [7].

2.2.2 Energy Consumption

Energy consumed by data centers is three times as much compared to the usage at current time [14]. The demand of energy will grow as more applications are now shifting towards cloud. Enormous amount of energy can be saved by sensible management of task at routers and base stations, mainly edge nodes [7].

2.2.3 Resource limitations

Smart phones and other user devices have low resource capabilities as compared to cloud devices [15]. Therefore, these devices send the data to cloud devices for better analytical computation. Data is relayed back and forth but all data is not desired by the services to perform. Hence, data can be analyzed and filtered at edge nodes [7].

2.2.4 Network Traffic

It is anticipated that with the increase of edge devices, generation of data will amount to almost 43 trillion gigabytes in the coming years which give rise to the problem of expanding data centers and heavy burden on network traffic. This again describes the potential of edge nodes and distribution of traffic along the nodes that are a hop or two away [7].

2.3 Challenges

Along with multiple opportunities provided by edge computing there are many challenges such as data privacy and security [12], deployment of workload, deployment strategies and policies involving connection [7]. Few important challenges at each layer that needs to be addressed are as follow:

2.3.1 Edge Node Computing

Several nodes between cloud and edge devices can facilitate edge computing such as gateways, routers, access points, aggregation points, switches and base stations [7] but not all nodes are available for computation mainly because of their own workload. To tackle this additional hardware service can be integrated such as Cisco's IOx[1] or we can upgrade existing resources for general purpose [16].

2.3.2 Offloading and Partitioning

Multiple partitioning techniques have been developed due to the evolution of distributed computing [17]. However, automated offloading is needed for computations without requiring explicit aptitude or location hence needing for schedulers to automate tasks at edge nodes.

[1] http://www.cisco.com/c/en/us/products/cloud-systems-management/iox/index.html

2.3.3 Service Quality

It is highly challengeable to ensure that the nodes are reliable and can manage as well as accommodate workload to attain high throughput. For this purpose, peak hours are determined to attain flexible schedule [18].

2.3.4 Privacy and Security

Data collected by sensors and other devices in any IoT infrastructure is very sensitive. Hence, privacy and security of data is a big issue even when edge computing is involved [12]. To protect data from malicious devices and attacks such as service manipulation and injection of false information, old security systems that only defend a described perimeter are not enough. New mechanism to survive internal and external attacks for devices whose functioning area is not known is required [6]. Many Trust based assessment models are designed to tackle this problem of identifying and separating malicious edge nodes along with solving the problem of authentication to some extent.

2.4 Edge Computing Applications

Various applications are in dire need of edge computing as compared to centralized computing such as smart city, smart grid, healthcare, data analytics [6]. Figure 2.2 shows some of the applications that need the support of services provide by edge computing.

2.4.1 Internet of Vehicles (IoV)

Apart from interconnection of IoV, connection is also made with infrastructure and Road Side Units (RSUs). For a large volume of vehicles RSUs provide real time services through distributed computing. Computation units are instilled in these vehicles to participate in the network of edge computing by two-way communication between edge servers deployed on RSUs and utilizing information processing in real time, mobility aware computation and autonomous driving [19].

2.4.2 Video Analytics

Independent sequence of events over watched by video cameras can be defined as video analytics. Most common example include autonomous video surveillance analysis. Traditional video surveillance system suffers from cloud computing analysis of

feeds because of privacy and high latency rate. With edge computing users can request edge servers for analytics distributed by cloud, hence, acquiring real time data [20].

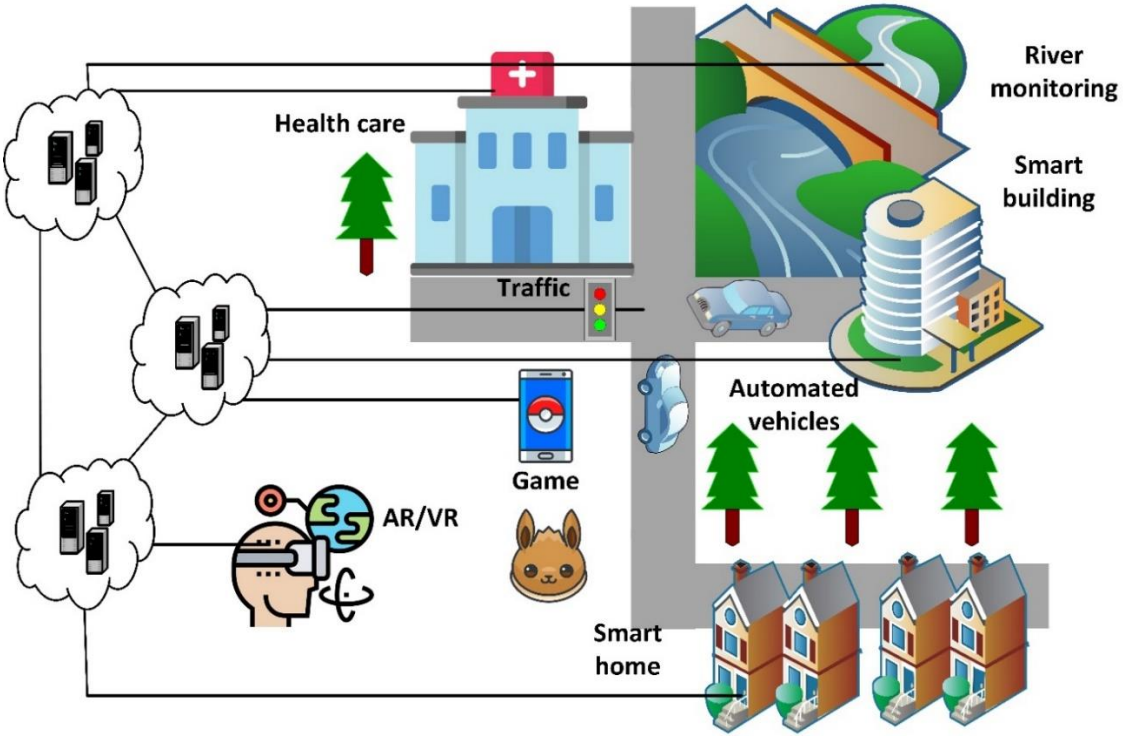

Figure 2.2: Edge Computing Applications[2]

2.4.3 Cloud Offloading

Devices like laptops, smart phones, smart TVs and wearables along with virtual reality (VR) and automated cars, there is a demand of low latency and real time analysis which is not possible through cloud centralized computing due to long distance between end users and cloud servers. With edge computing resources for analysis are available over the next hop and provide offloading of workloads and cashing of data. This is leveraging quality, improved latency, efficiency for time sensitive data [21].

2.5 Analysis of Existing Trust Models

Security risks such as reply attacks, message tampering and forging are always there to discourage users from utilizing edge nodes for computing. Hence a great urge to maintain trust of edge computing is there and many trust models are developed by scholars to tackle this problem. A detailed analysis of such models is shown in Table 2.1

[2] Some icons taken from www.flaticon.com Authors: Roundicons freebies, Eucalyp, Urban building.

Table 2.1: Analysis of Trust Models

Year	Authors	Paper Name	Mechanism of Model	Functions	Novelty
2018	Jie Yuan and Xiaoyong Li	A Multi-source Feedback based Trust calculation mechanism for Edge Computing	Rating generated through feedbacks by multiple resources mostly edge nodes.	Protection against bad mouthing attacks, suitable for large scale computing, increased reliability and speed.	Adoption of lightweight trust evaluation and introduction of broker layer.
2018	Yefeng Ruan, Arjan Durresi, Suleyman Uslu	Trust Assessment for Internet of Things in Multi-access Edge Computing	Trust framework based on measurement theory including confidence and trustworthiness.	Measurement of trust for nodes and applications, help in configuring resources and reduced redundancy.	Confidence is included to evaluate trustworthiness, introduction of trust assessment algorithm.
2018	Yousef Alsenani, Garth V. Crosby and Tomas Velasco	SaRa: A Stochastic Model to Estimate Reliability of Edge Resources in Volunteer Cloud	SaRa: A probability distribution model based on behavior of nodes.	Consider behavior and characteristics of tasks and better reliability.	Uses volunteer cloud for testing proposed model.

2017	Sandro Pinto, Tiago Gomes, Jorge Pereira, Jorge Cabral and Adriano Tavares	IIoTEED: An Enhanced, Trusted Execution Environment for Industrial IoT Edge Devices	By using trusted execution environments (TEEs) enabled by ARM TrustZone	Fulfilling IoT real time requirements and securing IoT edge devices	Evaluation of model achieved security in three fundamental domains of CIA
2017	Rafidha Rehmiman K A and Dr. S Veni	A Trust Management Model for Sensor enabled Mobile Devices in IoT	Trust protocols based on a central security manager and public key distribution.	Authentication, context aware location privacy, confidentiality, layer encryption and data origin authentication.	Introduces approach of public key for trust in IoT devices and protection on layer level.
2017	Mohamma d Alshehn and Farookh Khadeer Hussain	A Centralized Trust Management Mechanism for the Internet of Things (CTMIoT)	Trust management framework based using centralized system and super node.	Trust for multiple nodes in an IoT environment using super node as security manager	Centralized trust system for IoT
2017	Eungha Kim and Changsup Keum	Trustworthy Gateway System Providing IoT Trust Domain	Trustworthy gateway system based on trust domain in IoT	Protects from malicious attacks.	Using trust domain instead of traditional installation for security.

		of Smart Home			
2016	Sarah Asiri and Ali Miri	An IoT Trust and Reputation Model Based on Recommender Systems	Distributed probabilistic neural networks (PNNs)	Tackles cold start problem, considers sensitivity of data, better availability, fast response time.	Usage of neural networks to classify trustworthy nodes in a distributed setting.
2016	Caroline V. L. Mendoza and Joao H. Kleinschmidt	Defense for Selective Attacks in the IoT with a Distributed Trust Management Scheme	Distributed trust scheme by observing services of local nodes.	Prevention of selective attacks,	For performance analysis Contiki-OS is used.
2016	Sebastian Echeverria, Dan Klinedinst, Keegan Williams, Grace A. Lewis	Establishing Trusted Identities in Disconnected Edge Environments	Trust system based on secure key generation targeting non-centralized environment.	No dependence on central authentication, early distribution of credentials, any hardware security and tackling threats of a tactical environment	Use of key generation and distribution in a trust model without third party intrusion.
2016	Avani Sharma, Emmanuel S. Pilli,	A Framework to Manage Trust in	Trust management framework based on	Information gathering for trust computation,	Both centralized and distributed

	Arka P. Mazumdar, M. C. Govil	Internet of Things	machine learning, flow, fuzzy, probabilistic and statistical models.	dissemination and updation.	
2013	Jingpei Wang, Sun Bin, Yang Yu,Niu Xinxin	Distributed Trust Management Mechanism for the Internet of Things	Trust management based on formal semantics and fuzzy set.	Service, decision making and self-organizing	Early approach for layered distributed IoT
2001	Lalana Kagal, Scott Cost, Timothy Finin and Yun Peng	A Framework for Distributed Trust Management	Trust management scheme based on distributed systems	Manages permissions and delegations.	Re-delegation of tasks is addressed

J. Yuan et al. [8] introduced a trust model solely for serving edge computing that can be used for computing at a larger scale. Their multi-source feedback model adopts the idea of global trust degree (GTD) which are direct trust and feedback trust from brokers and edge nodes introducing three main layers: network, broker and device layer. Feedback is generated from edge devices as well as service brokers hence named as multi-source. Global convergence time (GCT) is used for the evaluation of efficiency. Experiments are conducted using NetLogo event simulator and Personalized Similarity Measure (PSM). To compute reliability Task Failure Ratio (TFR) is computed.

Y. Ruan et al. [6] proposed a trust model to evaluate applications as well as node in a network which additionally in real time help in resource configuration using measurement theory. Two metrices are defined; one to measure quality of probability termed as trustworthiness and to evaluate trustworthiness with measure error, confidence is introduced. In the framework multiple level of trust is taken into consideration such as:

trust of devices, tasks and device to device trust. Apart from this a new trust assessment algorithm is introduced to evaluate the model and a dynamic way to allocate resource for task using a trust threshold value and avoiding redundancy. Further results and implementation are not presented.

Y. Alsenani et al. [22] demonstrates a model, SaRa targeting volunteer cloud computing and in this scenario CuCloud which is a client/server architecture including volunteer machine and dedicated servers. It is a probabilistic model that is based on estimation of reliability of nodes by exploiting the behavior of nodes. Main parameters include task behavior and characteristics e.g. success, fail, priority etc. Though the model has random probability distribution, to validate this approach google clusters are used and testing environment contained hundreds of machines. As compared to other probabilistic models [23] SaRa has achieved greater precision.

There is a constant need that manufacturer of smart services provide configuration updates, control commands, send and receive status information. In this case Industrial IoT controllers need to protect themselves from unauthorized tempering and ensure accuracy of inputs. To tackle such problem S. Pinto et al. [24] have demonstrated a trust mechanism for Edge devices in industrial IoT environment to achieve confidentiality, integrity and authenticity at both hardware and software level. Since Trust Zone is gaining massive attention due to ARM processors so, the usage of Trust zone-based architecture is a sensible choice that implements trusted execution environment (TEE) in a slightly modified real time operating system (RTOS) but no further implementation is presented.

Mobile phones connected to wireless networks comprises more than half of IoT devices and these devices are vulnerable to many security threats. Therefore, R. Rehmiman et al. [25] 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 revolves 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, Elliptic Curve Cryptosystem are some of the techniques used by system for authentication along with public key generation and distribution by security manager. In addition, for packets anonymity and confidentiality, layer

encryption scheme and data origin authentication scheme are proposed in the model. The model is simple and addresses all challenges, but no proper demonstration and evaluation is carried out. Also, more than half of computation is carried by security manager which if fails brings down the whole system.

Trust management model based on centralized architecture for IoT is proposed by M. Alshehn et al. [26] that relies on a super node that works like that of a router and additionally monitors the whole network in a clustered environment head by master nodes, supervising cluster nodes. Super node consists of three modules: API module that provides interface between cluster nodes and master nodes for communication, Trust management module allows trust communication between super nodes, master nodes and cluster nodes by providing authentication data and Trust communication module allows two type of communication consisting of trust messages and value messages for establishing trust values. This model is unique for suggesting centralized approach, but it also comes with its disadvantages and the system is not implemented to prove its capability.

E. Kim et al. [27] proposed IoT trust domain to protect IoT infrastructure from malicious attacks through trustworthy gateway system. This system can be used for smart homes and can be extended to smart offices. The system uses home thing device sets IP address for source and destination and uses gateway system to pass the IP addresses, which then converted to ID's. ID table is used as a repository to store ID information of all connected devices in a network. Theoretically the system works fine as compared to untrusted domain, but no implementation is provided.

S. Asiri et al. [28] presented a model that uses distributed neural networks to classify trustworthy nodes. In this model they define Alpha nodes that are more capable control hubs and do not frequently change, responsible for managing jobs. Functionality of nodes is considered to make clusters. Type of data being transmitted is considered at profiling phase and used as one of the parameters for data security. Trustworthiness is determined based on threshold rating provided by nodes. Main phases include data collection, virtual clustering, weights calculation, transaction, trust computation, node classification and rate apprise. Though the system seems reliable no general demonstration of model was presented.

C. V. L. Mendoza et al. [29] offered a model that pointed out malicious nodes by the services they chose to provide. At first all nodes are assigned zero trust value and the process of neighbor discovery is started by sending announcement packets. When a node can provide a service, its trust value increases and if the node is not able to do so then its trust value decreases. This trust scheme is implemented in Cooja simulation provided by Contiki OS and detection of malicious nodes is successful.

To share information with one another, nodes share credential for verification involving third party intrusion. However, in a tactical environment such as search and rescue missions or military operations, access provision is limited. In addition, reliability on hardware and early share of credentials is not possible. To solve these problems S. Echeverria et al. [30] propose 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. Proposed trust solution uses Identity Based Cryptography (IBC), Stanford Identity Based Encryption (IBE) and OpenSSL ciphers. To evaluate the system, threat model by Microsoft SDL is used which propose 60 potential threats out of which 14 are considered for tactical environment. After implementation using open source tactical cloudlets 12 out of 14 threats were fully and partially handled.

A. Sharma et al. [31] proposed a generic trust management framework for IoT infrastructure. It defines all requirements to compute trust of edge devices with update and maintenance. A unique concept of trustor and trustee is considered to evaluate the system. Whole system consists of four phases. First phase gathers information through different parameters such as experience, reputation and knowledge. Models used for trust computation include machine learning, flow, fuzzy, probabilistic and statistical models. Two architectures centralized and decentralized are used for trust dissemination. Final phase includes update and maintenance which occurs in an event driven and time driven scenario. Since, it is a generic framework therefore no proper implementation is presented.

J. Wang et al. [32] defines trust mechanism as self-organizing items that take 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 sensor, core and application. Model uses

formal semantics-based language and fuzzy set theory to form trust. Results achieved consistent with an ideal situation and even though no demonstration was given for the working implementation of model, but it lays foundation for future models for IoT layered architecture

L. Kagal et al. [33] presents a trust management scheme that restricts re-delegation of task without following delegation protocol and deals with permissions in a distributed environment of supply chain management. For this purpose, CIIMPLEX EECOMS is chosen as an experimental environment and security agents are used for verification and authentication based on ID and verification certificates by Certificate Authority (CA) which further are used as tickets for access to resources. Permission to delegate a task to or by the agent is also given by security agents and log of delegations is maintained using Prolog. Delegations addresses are group, time bound, action restricted, strictly redelegate able and redelegate able delegation.

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 proposed frameworks is weak and, in some cases, not present. Moreover, Quality of Service (QoS) parameters used in the presented model propose novelty along with implementation in real time yielding better results.

2.6 Conclusion

In this chapter history of edge computing is introduced along with its necessity. Edge computing poses many challenges but also provides numerous functionality and applications. A detailed analysis of Trust management systems is conducted by describing their role and novelty at the end of this chapter.

C h a p t e r 3

PROPOSED FRAMEWORK

3.1 Introduction

This chapter includes discussion regarding the proposed trust management model for Edge computing devices. The proposed model will evaluate and establish trust between edge devices and edge servers, to provide trust and reliability of data and network. Following chapter discusses the proposed architecture and design of the trust management model.

3.2 Proposed Framework

The architecture of proposed trust management model is shown in figure 3.1.

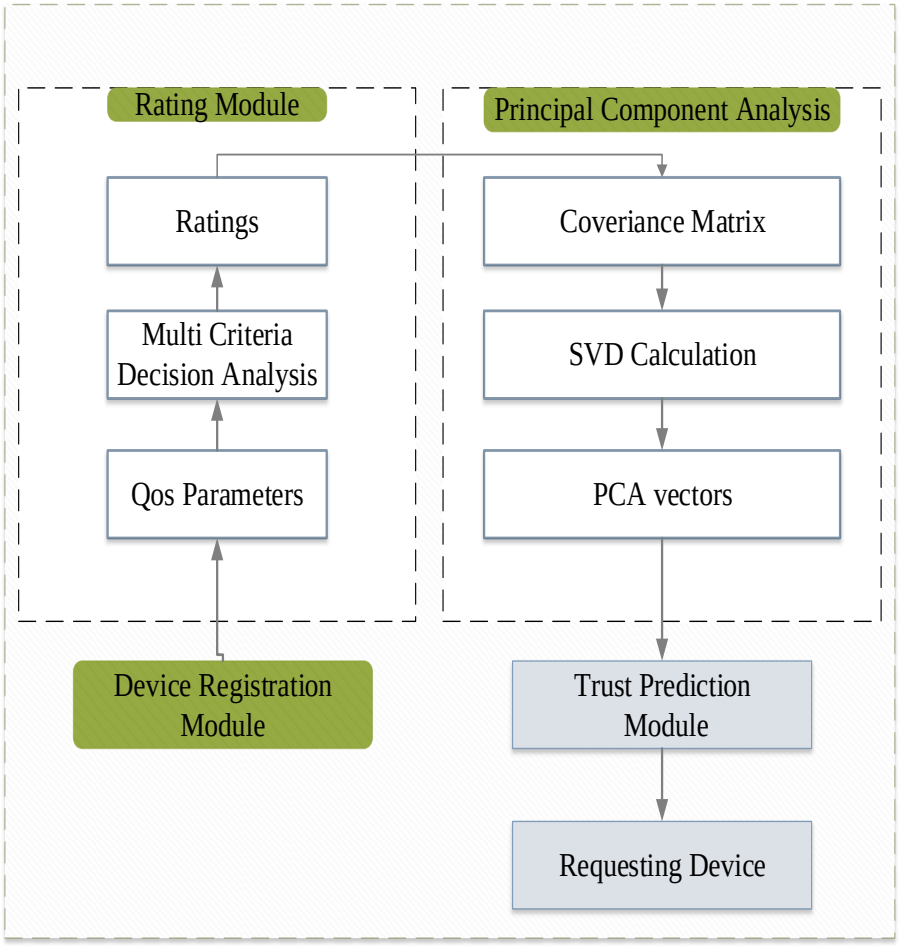

Figure 3.1: Architecture of Trust management model

The proposed model consists of two main modules, a rating management module and a trust calculation module. Rating module computes ratings based on QoS parameters

and multi criteria decision analysis which in turn produces covariance matrix, calculated SVD and PC vectors in trust calculation module.

Edge computing technology has vast applications, due to its ability to provide various benefits such as low latency, cloud offloading and saving bandwidth, it will be adopted globally 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, thus trust management becomes an important factor which provides some insight to the device, whose data is 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 or wrongful data to other devices or causing problems such as dos in the network, to reduce impact of such devices on the network, a lightweight trust management model is required which calculates trust based on the ratings of the device. The proposed model calculates the trust based on the ratings provided by the other devices, each device maintains its own ratings table for the devices it is communicating with, similarly edge servers or data centers also maintain a ratings table which store ratings from all the devices. Trust is calculated based upon those ratings, ratings in this model is based on following quality of service parameters.

An overview of the process is explained in a flowchart presented in Figure 3.2. In the first phase connection is established and communications starts between edge devices. Rating is calculated on the bases of these communications by computing covariance matrix and single vector decomposition. Trust is predicted and if the device is new and the cycle repeats for at least five times to calculate average trust for devices.

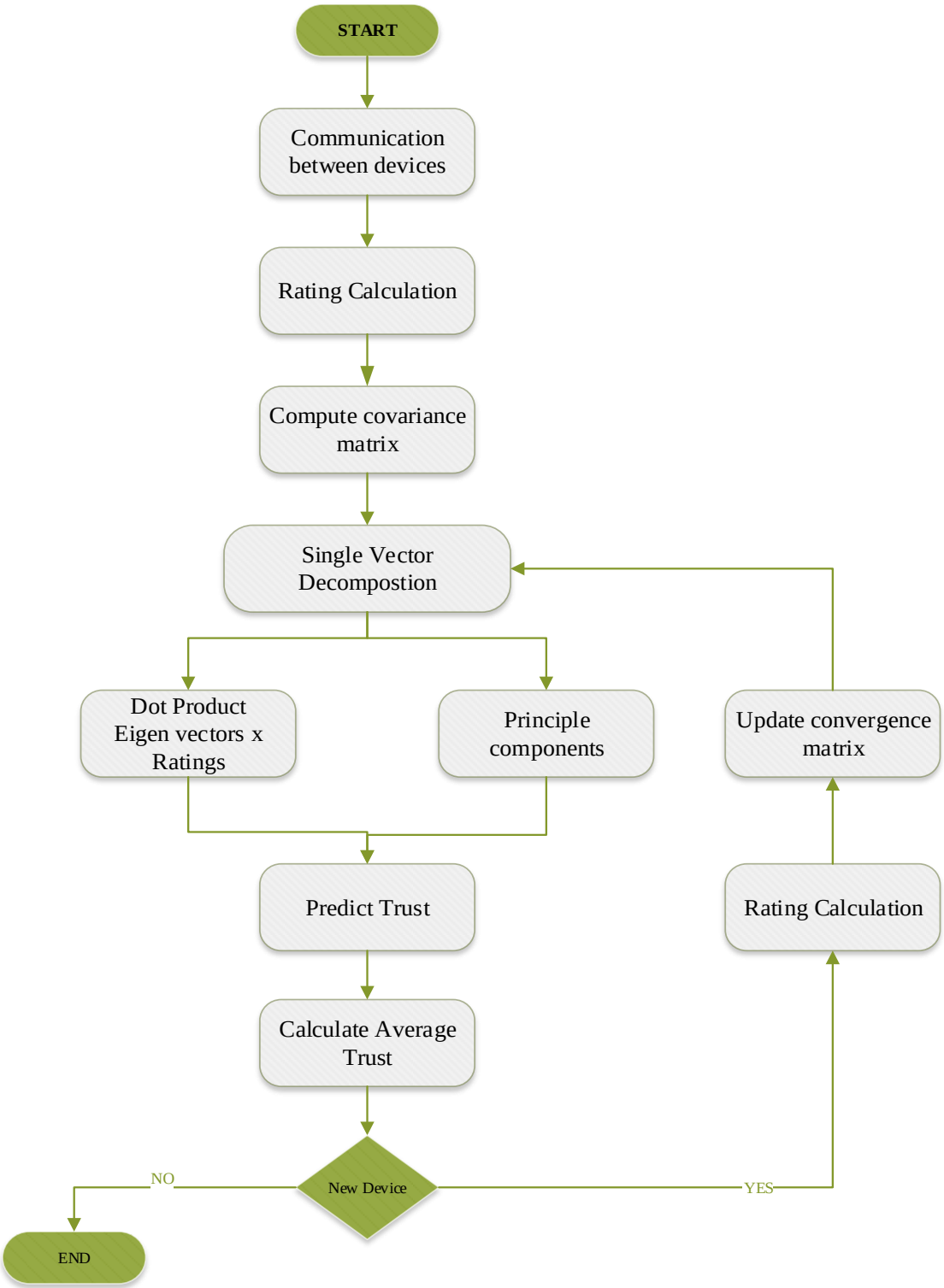

Figure 3.2: Flow diagram of proposed system

3.2.1 QoS Parameters

In 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 server device, and in other scenario where the device is gaining a service or data from another device it is categorized as client device, based on this criterion in our system each client device at the end of communications would provide feedback on QoS parameters.

Ratings are derived from these QoS parameters using multi-criteria decision analysis technique, and overall device trust is based upon these ratings. Trust in this model is an arithmetic value which lies in the range from 0 to 5, 5 being extremely trustworthy device and 0 being 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 the devices that are connected to an edge server are registered at by each edge server, and information is forwarded to the cloud. After registration, the system would be in observation mode, the observation mode would last for 1 transaction for each device. During these transactions following QoS parameters as given in Figure 3.3 would be recorded.

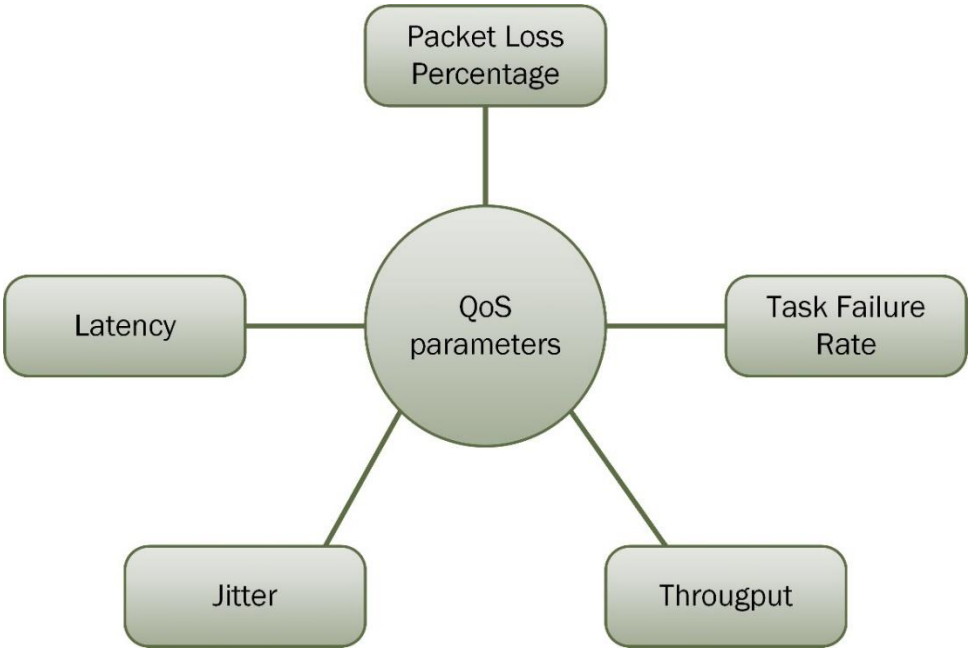

Figure 3.3: QoS parameters

3.2.1.1 Packet loss Percentage

It is defined as number of packets lost during the communication between two devices. High packet loss is considered bad for the network, and it would negatively affect 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 equation 3.1 [30].

$$PacketLossPercentage = \frac{\sum_{i=0}^{n} PL}{\sum_{i=0}^{n} PS} * 100 \quad (3.1)$$

Where, PL = Packets lost and PS = Total Packets sent.

3.2.1.2 Latency

Latency can be defined as the amount of time required for a packet to be transmitted from source to the destination. Latency is dependent on the congestion in the network, during the periods of high congestion latency increases causing low rating. it can be calculated using equation 3.2 [30].

$$Latency = \sum(PATime_i - PSTime_i) \quad (3.2)$$

Where, $PATime_i$ = Packet Arrival Time and $PSTime_i$ = Packet Send Time.

3.2.1.3 Jitter (Packet Delay)

The difference in the time between arrival of packets arriving at the destination in a particular time frame. It indicates the consistency and stability of the network. it can be calculated by equation 3.3 [30].

$$Jitter = \sum_{i=0}^{n} (\frac{Delay_i - Delay}{N}) \quad (3.3)$$

3.2.1.4 Throughput

Throughput is the number of bytes transferred from source to destination. It is measured in bits per seconds unit (bps) [30].

$$Throughput = \frac{\sum_{i=0}^{n}(Packets\ Received)}{\sum_{i=0}^{n}(StartTime - StopTime)} \quad (3.4)$$

3.2.1.5 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 applications that are being run on the network.

$$pft = \left(\frac{failed\ transactions}{total\ number\ of\ transactions}\right) \times 100 \quad (3.5)$$

3.2.2 Multi-criteria Decision Analysis

Multi criteria decision technique is used to take a decision of best selection among various options based on preferences of certain criteria. We explain MCDA in scenario of our implemented system.

3.2.2.1 Defining Criteria

Multiple quality of service parameters can be considered when communication is established between edge devices. Our experiment is based on determination of criteria shown in table 3.1.

Table 3.1: QoS parameters

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

Each parameter consists of some criteria that has a range of scores based on importance of resulting values [37].

Table 3.2: Score calculation criteria for QoS parameters

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

We are considering five parameters i.e. 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. Hence, we define good and bad criteria for defining these parameters as depicted in the table 3.2.

As it is shown in this table that those parameters which have inverse effect on performance of devices are already scored in inverse form, hence the beneficial criteria do not need to me divided with minimum value to make all parameters comparable. The measurement criteria presented in this table is based on research [35].

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, latency are non-beneficial criteria in contribution of ratings, so their minimum values achieve highest score. Whereas throughput is a beneficial parameter hence its higher values get maximum score. The scoring is allotted from 1 to 5 because we have 5 parameters and want to get final rating up to 5. For alternate scenarios where 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 same unit; this keeps us from taking a range of values and normalizing it to get weighted normalized decision matrix.

Rating is denoted as:

$$R_i = \sum_{j=1}^{n} w_{ij}\, a_{ij} \tag{3.6}$$

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

3.2.2.2 Methodology

Methodology for obtaining ratings from given QoS parameters is as follows.

Step 1: Determining Alternatives

Values of QoS parameters are obtained through simulation as represented in table 3.3. The values of each device are portrayed such as all the scenarios are covered ranging from best case scenario to worst case scenario of scores.

Table 3.3: QoS Parameters

QoS Parameters	Throughput	Latency	Jitter	Packet loss percentage	Task Failure Ratio
Device 1	1 Mbps	280 ms	35 ms	35%	55 tasks
Device 2	25 Mbps	175 ms	25 ms	17%	40 tasks
Device 3	55 Mbps	75 ms	15 ms	12%	20 tasks
Device 4	75 Mbps	35 ms	7 ms	7%	7 tasks
Device 5	95 Mbps	25 ms	2 ms	3%	3 tasks

List of Devices ↓

Step 2: Assigning Weight

We assign relative weight to each parameter based on their importance in given scenario. Weight of each QoS parameters can be assigned as per 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.

$$\sum_{i=1}^{n} w_i = 1 \tag{3.7}$$

Table 3.4: Weightage assigned to each QoS Parameter

Weightage →	**0.20**	**0.20**	**0.20**	**0.20**	**0.20**
Parameters →	Throughput	Latency	Jitter	Packet loss percentage	Task Failure Ratio

We assign more weightage to those parameters which hold strong position in evaluation of trust among devices. The sum of weightage is always 100 percent. In this scenario, we have assigned average weight i.e. 0,20 to all parameters.

Step 3: Value of scores.

The parameters of each device are assigned score based on its values recorded during the communication session.

Table 3.5: Score of each device based on QoS parameters

QoS Parameters ⟶	Throughput	Latency	Jitter	Packet loss percentage	Task Failure Ratio
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

Step 4: Final Score

Multiply weight assigned to each parameter with its score.

$$\boldsymbol{R_i = w_i a_i} \tag{3.8}$$

Table 3.6: Value of Scores

QoS Parameters ⟶	Throughput	Latency	Jitter	Packet loss percentage	Task Failure Ratio
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

Step: 5 Final Ratings

Final ratings for each device are obtained by sum of all final scores of QoS parameters of a device.

$$\boldsymbol{R_{ij}} = \sum_{j=1}^{n} w_{ij}\, a_{ij} \tag{3.9}$$

$$\boldsymbol{R_{ij}} = \boldsymbol{w_t}(\boldsymbol{T_{ij}}) + \boldsymbol{w_p}(\boldsymbol{P_{ij}}) + \boldsymbol{w_j}(\boldsymbol{J_{ij}}) + \boldsymbol{w_t}(\boldsymbol{L_{ij}}) + \boldsymbol{w_f}(\boldsymbol{F_{ij}}) \tag{3.10}$$

Table 3.7: Rating of devices

QoS Parameters →	Throughput	Latency	Jitter	Packet loss percentage	Task Failure Ratio
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 3.8 shows final ratings obtained for each device. It is observed that devices with lower score have low ratings whereas devices with higher score have high rating.

Table 3.8: Final Ratings

List of devices	Final Ratings
Device 1	0.5
Device 2	2
Device 3	3
Device 4	4
Device 5	5

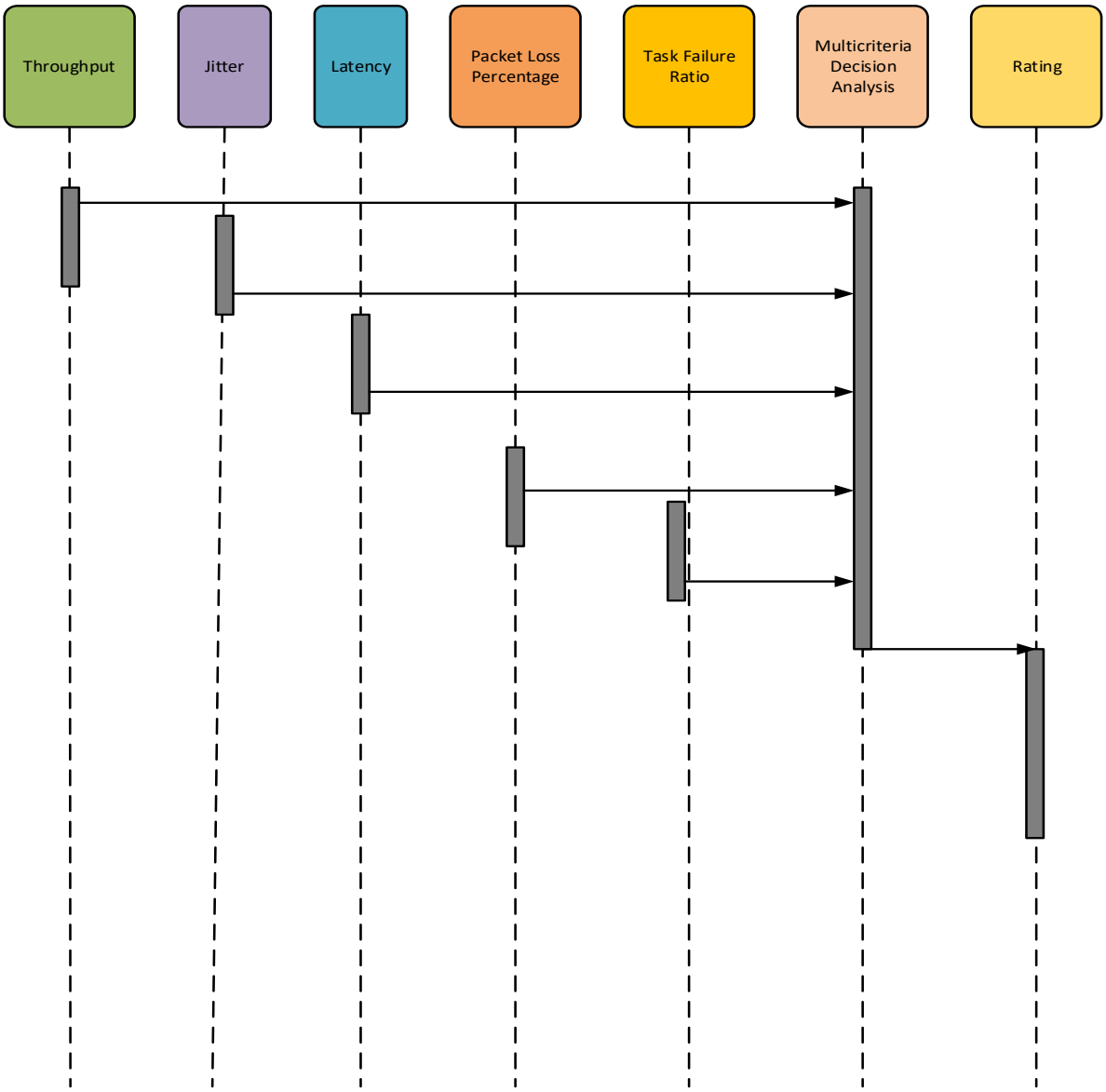

Figure 3.4: Sequence diagram of Rating Module

Figure 3.4 shows continuation of Trust Computation where after calculation throughput, jitter, latency, packet loss percentage and task failure ratio weights are assigned based on preference or requirement of the network. After weight assignment multi-criteria decision analysis approach is used to calculate the overall ratings of devices. If two d_i and d_j were to be computed the resultant Rating would be known as R_{ij}. R is then forwarded to our trust calculation matrix, where it is converted into a symmetric matrix and svd is calculated, using that svd new matrix is reconstructed using the most significant eigen values. To predict trust based on ratings.

3.2.3 Algorithm

Algorithm 1 Trust Calculation Procedure: Trust

Require: S: a stream of examples, R= Ratings of devices (from 0 to 5)
RM = Rating matrix

1: Network Establishment
2: Begin Procedure Trust
3: A stream of examples S arrives
4: Multi criteria Decision Analysis (T,J,L,P,F)
6: If Cold start = yes then
7: Single Vector Decomposition (RM)
8: else
9: Incremental Singular Value Decomposition
11. End if
12: Matrix Reconstruction
13: END Procedure

Algorithm 1 is the main algorithm, where all the functionalities are defined, and control is returned after execution.

Algorithm 2 Trust Calculation Procedure: Multi criteria Decision Analysis

1: Define Quality of service parameters
T: Throughput
J: Jitter
L: Latency
P: Packet loss
F: Packet Failure Rate (pfr)

```
3: Latency(val){
        if (val < 30){
        return 5;
        } else if(val >= 30 and val <50){
         return 4;
        } else if(val >= 50 and val <150){
        return 3;
        } else if(val >= 150 and val <250){
        return 2;
        } else {
        return 1;
   }
4: Packetloss(val){
        if (val < 4){
        return 5;
        } else if(val >= 4 and val <10){
        return 4;
        } else if(val >= 10 and val <15){
        return 3;
        } else if(val >= 15 and val <20){
        return 2;
```

```
            } else {
            return 1;
   }
5: Taskfailure (val){
            if (val < 5){
            return 5;
            } else if(val >= 5 and val <10){
            return 4;
            } else if($val >= 10 and val <30){
            return 3;
            } else if($val >= 30 and val <50){
             return 2;
            } else {
            return 1;
   }
6: Jitter(val){
            if (val < 5){
            return 5;
            } else if(val >= 5 and val <10){
            return 4;
            } else if(val >= 10 and val <20){
            return 3;
            } else if(val >= 20 and val <30){
            return 2;
            } else {
             return 1;
   }
7: throughput(val){
            if (val < 30){
            return 1;
            } else if(val >= 30 and val <50){
             return 2;
             } else if(val >= 50 and val <70){
             return 3;
             } else if(val >= 70 and val <90){
            return 4;
            } else {
             return 5;
   }
8: Assign weights to all parameters
9: return value of parameters = score
10:  Final Score = R_i = w_i a_i
11:  Final Ratings = R_ij = sum_{j=1}^{n} w_ij a_ij
12: return R_ij
```

Algorithm 2 is of multi-criteria decision analysis; we are taking all the parameters as input. Defining our criteria to assign score, final rating is derived by using Weighted sum model,

Algorithm 4 Trust Calculation Procedure: Single Vector Decomposition (RM)

```
1. Function cal_svd(t1)
        B = transpose(t1); transposee
         C = t1*B;              coverience Matrix
        [U, S, V] = svd(C)      Eigen vectos
End
```

Algorithm 4 is a function which calculates the SVD. It takes Ratings as input and returns three matrix U, S ,V

Algorithm 5 Trust Calculation Procedure: Incremental Singular Value Decomposition

```
1. Function Incremental_svd (U, S, V, t1, R)
        new=[t1; R]
        [Up, Sp, Vp] = svd_update(U, S, V, R, true);
 End
```

Algorithm 5 uses an implemented function svd_update which takes arguments, of previous svd and new single rating string and calculates the new SVD.

Algorithm 6 Trust Calculation Procedure: Matrix Reconstruction

```
Function reconstruction (t1, U, devices)
        for b=1: devices
                for a=1: devices
                        predicted_value=U(:,a);
                        previous_rating(numel(predicted_value)) = 0;
                         a1(b,a)= dot(t1 (b,:),transpose(predicted_value)); Dot Product all
        ratings Eigen Vectors
                end
        end
        Dot Product all ratings Eigen Vectors
        for k=1: devices
                pr=0;
                        for c=1:5
                                pr1=a1(k,c)*U(:,c);
                                pr= pr+pr1;
                        end
                pl(:,k)=pr;  predicted Trust
        end
        return pl
end
```

Algorithm 6 recreates the matrix and returns the predicted trust value.

3.2.4 Single Value Decomposition

It is a matrix factorization technique, used mainly for dimensional reduction. It is used to reduce the dimensionality 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 user preference of the past [31]. In our system we use collaborative filtering to find out trust of the 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 maintain that either the device is good or bad based on 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.

$$\boldsymbol{A = USV^T} \tag{3.11}$$

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.

Step 1: Covariance Matrix

Convergence matrix is calculated from combining the rating vectors. This step helps identify how variables are correlated. The convergence matrix is a symmetric matrix. In order to achieve this symmetry following formula is utilized

$$\boldsymbol{A = A \times transpose(A)}$$

Let

$$\text{A} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 1 & 0 \end{bmatrix} \Rightarrow AA^T = \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 1 \end{bmatrix} = \text{B} \tag{3.12}$$

Step 2: Compute eigenvalues of A

For a square matrix B of order x, the number $\boldsymbol{\lambda}$ is an eigen value if and only if there exists a non zero matrix C such that BC = $\boldsymbol{\lambda}$C.

$$\text{As} \quad \boldsymbol{B\,x\,\lambda = \lambda\,x} \text{ so } \mathbf{(B - \lambda I)\,x = 0} \tag{3.13}$$

Equation 3.13 is called characteristic equation of B, and is an n^{th} order polynomial in $\boldsymbol{\lambda}$ with n roots. These roots are called the eigenvalues of B.

Step 3: Matrix Reconstruction

The most significant eigen vectors are utilized to construct the final matrix, this matrix represents the final predicted trust value. If prediction depends upon criteria, how many of the generated eigen vectors are to be utilized, as eigen vectors are arranged in descending order. The first number shows highest significance, then the rest of the values. During the matrix reconstruction.

Dot Product all ratings Eigen values and ratings is calculated

$$\boldsymbol{A = Dot(ratings, transpose(U))} \quad (3.14)$$

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

$$\boldsymbol{PrT = [A_{mxn}][U_n]} \quad (3.15)$$

At the end of the process, 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.

3.2.5 Incremental Singular Value Decomposition

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 threshold value 2.5. it is considered as 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 Value Decomposition algorithm. Incremental SVD algorithm would find it's trust based on communications it had with other devices [34].

3.3 Conclusion

This chapter concludes the technique which has been improvised in order to calculate the trust of edge devices. It discusses the architecture along with working flow of proposed Trust Management Model is described in detail. To calculate ratings for trust calculation of edge devices Quality of Service (QoS) parameters are introduced. Using

latency, packet loss percentage, jitter, throughput and task failure rate as QoS parameters, weights are assigned to devices using single vector decomposition by calculating covariance matrix and computing eigen values. Experimental implementation of our technique has been presented in the next chapter.

C h a p t e r 4

IMPLEMENTATION AND RESULTS

4.1 Introduction

This chapter contains the details regarding our proposed trust management framework for edge computing. 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 trust calculation module we are using Singular value decomposition. Trust is derived directly from ratings. For new devices on the network incremental SVD algorithm is employed.

4.2 Experimental Setup

Our proposed system is implemented on MATLAB, because of limitation of resources, and availability of edge network simulators. For simulating 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. After extraction of required parameters from EdgeCloudSim we are storing them in MySQL database, which could be accessed by using http://localhost/phpmyadmin. PhpMyAdmin can be enabled by installing xampp in windows which provide a local server and a database server, we use those values to calculate the QoS parameters and derive our ratings by employing multi-criteria decision analysis technique. We are calculating our ratings in a php web project. After calculations of our ratings, we are then exporting those ratings to a csv file, from which information can be imported into MATLAB, we are using MATLAB because MATLAB natively supports matrix functions such as transpose and SVD.

4.2.1 Implementation of Trust Management system for edge devices

Figure 4. shows our devised system of edge devices network established. At first, we have computed trust for 10 devices and 100 after that. 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 in 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 power of server devices, that after every service provided by the server device, it is being rated based on its QoS

parameters. These QoS parameters are translated into 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 such that

$$\mathbf{R= R.transpose(R)} \tag{4.1}$$

This would result in a square matrix. This is a necessary requirement to find out 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.

$$\boldsymbol{A = USV^T} \tag{4.2}$$

Here A is a $X \times Y$ matrix, U is an $X \times Y$ orthogonal matrix whose transpose is equal to its inverse. $U^{-1} = U^T$.

S is an $X \times X$ diagonal matrix whose all values other than diagonal are zero.

V is an $X \times X$ orthogonal matrix. $V^{-1} = V^T$

4.3 Example Scenario:

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

4.3.1 Rating Matrix

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

$$\boldsymbol{R_{ij} = \sum_{j=1}^{n} w_{ij}\, a_{ij}} \tag{4.3}$$

This gives us a 9×10 rating matrix for 10 devices. 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.

$$\mathbf{R\ (10 \times 10)} = \mathbf{R}\ \boldsymbol{R^T} \tag{4.4}$$

1	2	3	4	5	6	7	8	9	10
3.8000	3.2000	2.8000	0	3.4000	4	3.4000	3.4000	2.4000	4.8000
3.8000	3.8000	2.8000	3.2000	3.2000	0	3.8000	3	3.6000	2.6000
4.6000	4.4000	4	0	4.8000	3.2000	3.2000	0	3	4
2.8000	0	3.4000	4	3.2000	3.8000	2.8000	3	3.8000	0
3.2000	3.4000	3.2000	0	2.8000	2.6000	3.4000	0	3.6000	3.8000
3.2000	3.2000	4	2.6000	4.6000	4.8000	2.6000	4.2000	3.2000	2.2000
2.6000	3.8000	4.4000	0	3.2000	3.2000	4.4000	3	3	3.2000
3.6000	0	4.6000	3.2000	0	4.4000	2.8000	4.2000	0	2.8000
3.4000	4.2000	0	2.6000	2.8000	0	3	4.2000	0	4

Figure 4.1: Rating matrix

Figure 4.1 shows the ratings which have been allotted from one device to another. The highlighted illustrates that these spaces are left blank because some devices have not rated others.

4.3.2 Singular Value Decomposition:

Singular value decomposition technique is used to generate 3 other matrices from R.

$$\mathbf{R} = \boldsymbol{USV^T} \tag{4.5}$$

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.

$$\mathbf{R}\ \boldsymbol{R^T} = \boldsymbol{USV^T}(\boldsymbol{USV^T}) = \boldsymbol{US^2V^T} \tag{4.6}$$

$$\mathbf{R}\ \boldsymbol{R^TV} = \boldsymbol{VS^2}$$

Equation 4.6 is equal valent to eigenvector definition for the matrix. R R^T is the matrix, V contains all eigenvectors and VS^2 contains all eigen values. The figures below show experimental results from our implemented system.

	1	2	3	4	5	6	7	8	9	10
1	-0.3761	0.0136	-0.0316	-0.1048	-0.4955	0.1549	0.2365	0.1721	0.6849	-0.1489
2	-0.3245	-0.4904	-0.2858	-0.0809	0.0624	0.1165	-0.1063	-0.4108	0.0950	0.5995
3	-0.3580	0.1914	0.3655	0.1666	0.3790	0.6387	0.3213	-0.0701	-0.1166	0.0233
4	-0.1755	0.5462	-0.3538	0.2101	-0.5147	0.1417	-0.0852	-0.1201	-0.3923	0.2023
5	-0.3477	-0.2401	-0.1270	0.3622	-0.0067	-0.4635	0.5376	-0.1694	-0.2422	-0.2878
6	-0.2568	0.2339	0.6702	-0.2549	-0.1569	-0.4543	-0.0460	-0.2073	-0.0320	0.2969
7	-0.3578	-0.0086	0.0027	-0.0467	0.0608	0.0556	-0.5957	-0.3904	0.0023	-0.5964
8	-0.2954	0.4737	-0.4145	-0.2388	0.5462	-0.2928	0.0253	0.1647	0.2021	0.0832
9	-0.2808	-0.1128	0.1431	0.6155	0.0599	-0.1269	-0.4187	0.5197	0.0772	0.1951
10	-0.3366	-0.2735	-0.0322	-0.5257	-0.1084	0.0905	-0.0190	0.5108	-0.4979	-0.0730

Figure 4.2: Experimental Results U

Figure 4.2 illustrates that U is an orthogonal matrix which is generated as a result of decomposition of rating matrix R shown in figure 4.1.

	1	2	3	4	5	6	7	8	9	10
1	755.0756	0	0	0	0	0	0	0	0	0
2	0	58.3907	0	0	0	0	0	0	0	0
3	0	0	45.5991	0	0	0	0	0	0	0
4	0	0	0	31.1627	0	0	0	0	0	0
5	0	0	0	0	7.7302	0	0	0	0	0
6	0	0	0	0	0	6.2386	0	0	0	0
7	0	0	0	0	0	0	4.4993	0	0	0
8	0	0	0	0	0	0	0	1.5267	0	0
9	0	0	0	0	0	0	0	0	0.1370	0
10	0	0	0	0	0	0	0	0	0	3.8593e-16

Figure 4.3: Experimental results S

Figure 4.3 shows that S is a diagonal matrix which has entries only along the diagonal. It contains square roots of all eigenvalues of R R^T.

	1	2	3	4	5	6	7	8	9	10
1	-0.3761	0.0136	-0.0316	-0.1048	-0.4955	0.1549	0.2365	0.1721	0.6849	0.1489
2	-0.3245	-0.4904	-0.2858	-0.0809	0.0624	0.1165	-0.1063	-0.4108	0.0950	-0.5995
3	-0.3580	0.1914	0.3655	0.1666	0.3790	0.6387	0.3213	-0.0701	-0.1166	-0.0233
4	-0.1755	0.5462	-0.3538	0.2101	-0.5147	0.1417	-0.0852	-0.1201	-0.3923	-0.2023
5	-0.3477	-0.2401	-0.1270	0.3622	-0.0067	-0.4635	0.5376	-0.1694	-0.2422	0.2878
6	-0.2568	0.2339	0.6702	-0.2549	-0.1569	-0.4543	-0.0460	-0.2073	-0.0320	-0.2969
7	-0.3578	-0.0086	0.0027	-0.0467	0.0608	0.0556	-0.5957	-0.3904	0.0023	0.5964
8	-0.2954	0.4737	-0.4145	-0.2388	0.5462	-0.2928	0.0253	0.1647	0.2021	-0.0832
9	-0.2808	-0.1128	0.1431	0.6155	0.0599	-0.1269	-0.4187	0.5197	0.0772	-0.1951
10	-0.3366	-0.2735	-0.0322	-0.5257	-0.1084	0.0905	-0.0190	0.5108	-0.4979	0.0730

Figure 4.4: Experimental results V

Figure 4.4 shows matrix V, which is also an orthogonal matrix and contains eigen vectors of R R^T as described previously in this section.

1	2	3	4	5	6	7	8	9	10
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

Figure 4.5: Predicted Trust after matrix reconstruction

Figure 4.5 shows the predicted values where, ratings were not assigned, highlighted are the new predicted trust values.

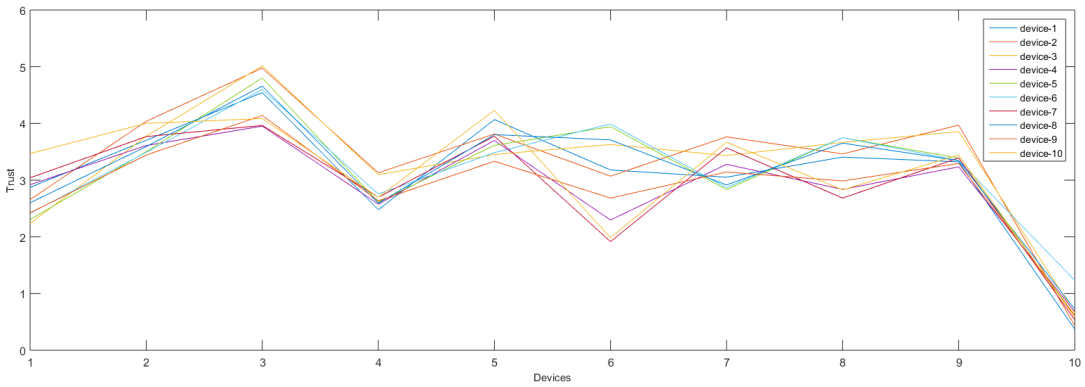

Figure 4.6: Trust calculation graph

Figure 4.6 represents experimental results of trust calculation in our simulation system. These results were generated from initial step of including 10 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 value. These results support our study explained in chapter 3 of this thesis that those devices which had high ratings based on quality of service parameters have turned out to be more trustworthy as compared to low rated devices which had gained less score in the initial steps and have yielded lower values of trust.

4.3.3 Incremental Singular Value Decomposition

The established network up to now has been observing a network with 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. Off course, a whole new system cannot be established from scratch to observe the trust level of this new device k. Therefore, we have implemented the technique of incremental SVD for predicting the trust of latest added devices to the network. This method is a continuity of SVD which we have presented previously.

$$\mathbf{R} = \boldsymbol{USV^T}$$

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

Let

$$\boldsymbol{Z} = \frac{\boldsymbol{U}}{\boldsymbol{R}} = \boldsymbol{U^T R} \tag{4.7}$$

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

Let

$$\mathbf{H} = (\mathbf{I} - \mathbf{U}\boldsymbol{U}^{\boldsymbol{T}})\mathbf{R} = \mathbf{R} - \mathbf{UZ} \tag{4.8}$$

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

Let

$$\mathbf{X} = \frac{\boldsymbol{K}}{\boldsymbol{H}} = \boldsymbol{K}^{\boldsymbol{T}}\boldsymbol{H} \tag{4.9}$$

In equation 4.10 K is orthogonal basis of H and X is the projection of R onto the space orthogonal to U.

Consider the following example,

$$[\boldsymbol{U}\ \boldsymbol{K}]\begin{bmatrix}\boldsymbol{diag}(\boldsymbol{s}) & \boldsymbol{Z}\\ \mathbf{0} & \boldsymbol{X}\end{bmatrix}\begin{bmatrix}\boldsymbol{V} & \mathbf{0}\\ \mathbf{0} & \boldsymbol{I}\end{bmatrix}$$

$$= [\boldsymbol{U}\ (\boldsymbol{I} - \boldsymbol{U}\boldsymbol{U}^{\boldsymbol{T}})\boldsymbol{R}/\boldsymbol{K}]\begin{bmatrix}\boldsymbol{diag}(\boldsymbol{s}) & \boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{R}\\ \mathbf{0} & \boldsymbol{K}\end{bmatrix}\begin{bmatrix}\boldsymbol{V} & \mathbf{0}\\ \mathbf{0} & \boldsymbol{I}\end{bmatrix}$$

$$= [\boldsymbol{U}`\ \boldsymbol{diag}\ (\boldsymbol{s})\boldsymbol{V}^{\boldsymbol{T}}\boldsymbol{C}] = \mathbf{[M\ R]} \tag{4.10}$$

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,

$$\mathbf{U'\ diag}\ (\boldsymbol{s}`)\boldsymbol{V}`^{\boldsymbol{T}} \leftarrow^{\boldsymbol{SVD}} \mathbf{D} \tag{4.11}$$

```
p1 =

    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
```

Figure 4.7: Predicted Trust P1 given by new device

Figure 4.7 shows the addition of a new device, and it's predicted ratings for the devices it communicated with. The accuracy of these ratings is directly proportional to the number of devices communicating with each other in the network

4.4 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

4.4.1 Average Ratings and Average Trust

Average rating is calculated for devices according to the methodology discussed in previous chapter.

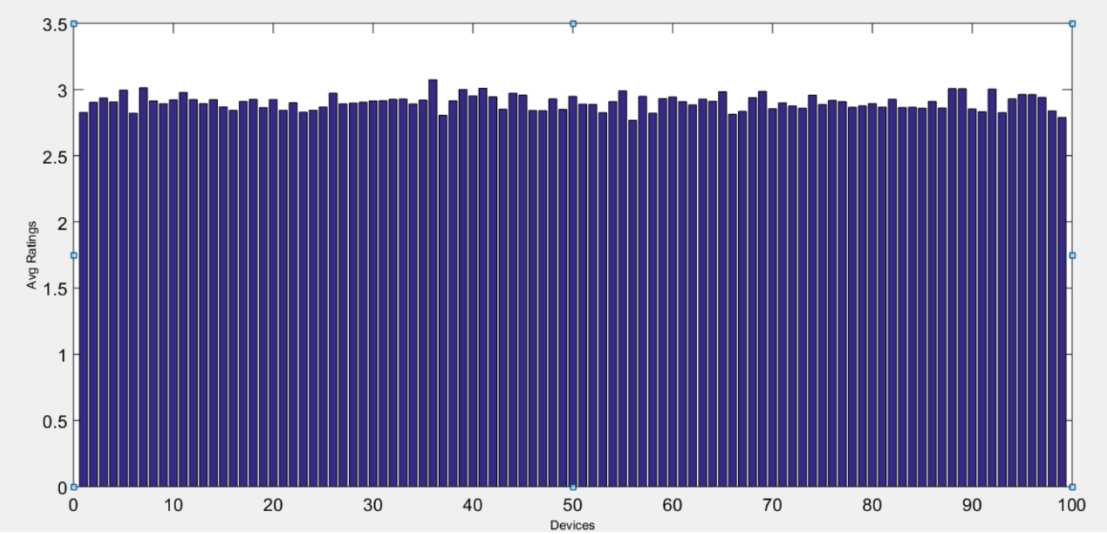

Figure 4.8: Average Ratings Graph

As presented in the figure 4.8. devices with high score in QoS parameters have high average ratings, while devices with low QoS parameters have low average ratings.

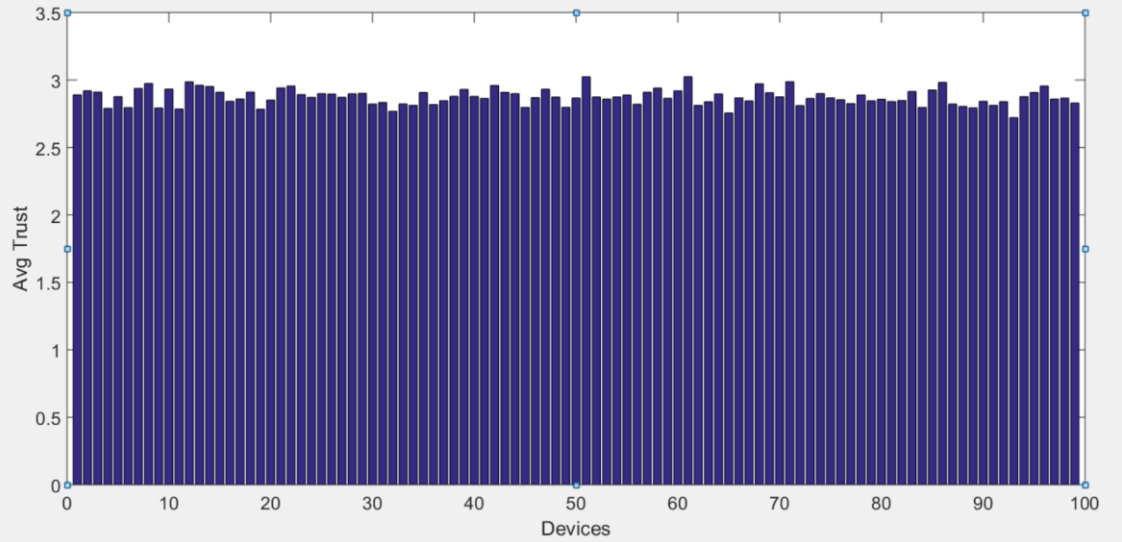

Figure 4.9: Average Trust Graph

Average trust graph is shown in figure 4.9. The trust was computed according to the criteria previously presented in this research. This graph reflects that rating of device can be higher but due to community effect it's trust can have lower value. Community effect is the input from all the other devices, taken into consideration during trust calculation.

4.4.2 Ratings Scatter Graph and Trust Scatter Graph

The image represents ratings scatter graph for 1000 devices in a network. Most of the ratings lie at average distance from each other. When comparing both graphs in figure 4.10 and 4.11 we observe that devices have both positive correlation and negative correlation, and in 4.11 due to community effect the ratings spread is much less.

]

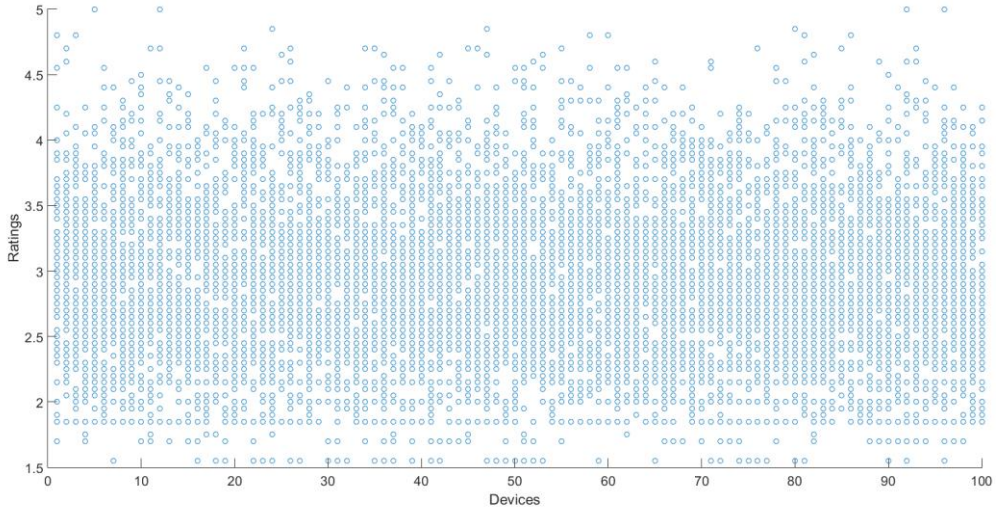

Figure 4.10: Ratings Scatter graph

Figure 4.10 represents the ratings of each device in scattered form. It shows maximum, minimum and concentration of ratings between 1 and 5 for each of 100 devices, as per their communication.

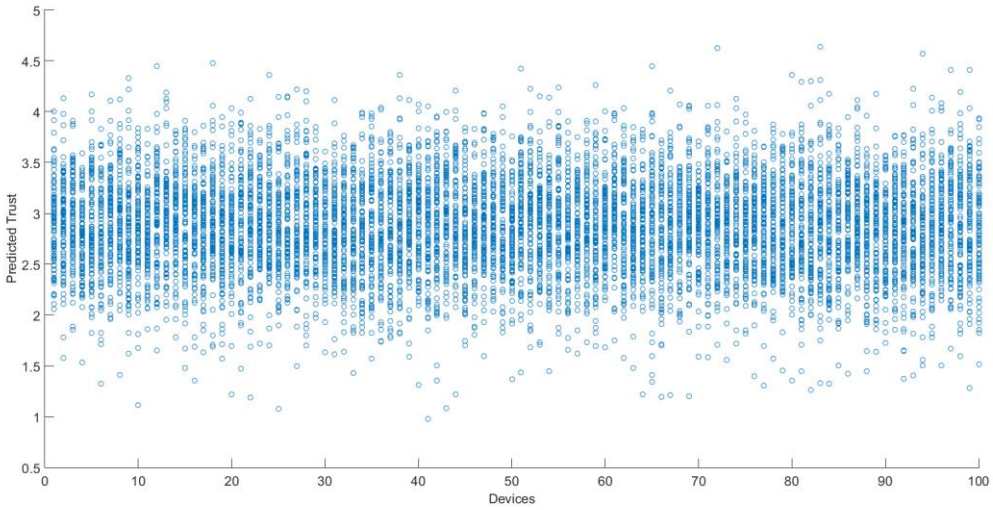

Figure 4.11: Trust Scatter Graph

Figure 4.11 shows the predicted trust range, maximum, minimum values and concentration of trust values for each device. Due to community factor, it is not essential for devices having higher ratings to have high trust as well. For example, in figure 4.10 a device may have a single rating of 4 or 4.5 but as we attain the predicted trust, the result may be lying at average trusted value of 2.5 to 3.

4.5 Conclusion

In this chapter proposed Trust Management Model is implemented using MATLAB, EdgeCloudSim, MySQL database and results are evaluated. SVD model used for decomposing ratings matrix is discussed in detail and an example scenario is presented to calculate rating matrix. At the end improved results of experiment for 100 devices is presented.

C h a p t e r 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

This chapter concludes the work of presented thesis and lay out some of the future research foundations for trust management models in edge computing. It also sheds some light on the aim of this research, its goals and further improvements.

Edge computing as we know today is an emerging technology where generation, distribution, storage and computation of data is performed at the edge of the network. The 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 security and reliability of edge devices by proposing a trust management model to evaluate the credibility of edge nodes.

Research include history, basic concepts, evolution, integration and adaptability of Edge Technology. Trust management models and their comparative analysis along with challenges, need for trust management and advantages along with the proposed framework for trust management system. The architecture, its individual modules, implementation technique and key components. Execution mechanism of the proposed trust management system have been discussed with fruitful results

The proposed model calculates the trust based on the ratings provided by the other devices, each device maintains its own rating table for the devices it is communicating with, similarly edge servers or data centers also maintain a rating table which store ratings from all the devices. Trust is calculated based upon those ratings based on quality of service parameters such as packet loss percentage, latency, jitter, throughput and task failure ratio. Each parameter consists of some criteria that has a range of scores based on importance of resulting values and weight is assigned to the devices accordingly.

5.2 Future work

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. Future research can include

smart service level agreement where nodes selectively decide what kind of service they will provide to the devices on the network, to save resources and since the implementation of presented model is evaluated using MATLAB it can further be implemented on a better edge n. Current work is limited to QoS parameters of the networks we would like to extend our work and increase the QoS parameters and include QoS parameters of application or data example using hash to compare correctness of data being recorded and sent.

REFERENCES

[1] Ren, J., Pan,., Goscinski, A., Beyah, R. A., 2018, "Edge computing for the Internet of things", IEEE Network, 32(1), 6-7

[2] Hong, H. J., 2017, "From cloud computing to fog computing: unleash the power of edge and end devices", IEEE international conference on cloud computing technology and science (CloudCom), pp. 331-334

[3] Panchali, B., 2018, "Edge Computing-Background and Overview", IEEE International Conference on Smart Systems and Inventive Technology (ICSSIT), pp. 580-582

[4] Liu, K., Gurudutt, A., Kamaal, T., Divakara, C., Prabhakaran, P., 2017, "Edge computing framework for distributed smart applications". IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), pp. 1-8

[5] Garcia Lopez, P., Montresor, A., Epema, D., Datta, A., Higashino, T., Iamnitchi, A., Riviere, E., 2015, "Edge-centric computing: Vision and challenges"., ACM SIGCOMM Computer Communication Review, 45(5), pp. 37-42.

[6] Ruan, Y., Durresi, A., Uslu, S., 2018, "Trust Assessment for Internet of Things in Multi-Access Edge Computing", IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA), pp. 1155-1161

[7] Varghese, B., Wang, N., Barbhuiya, S., Kilpatrick, P., Nikolopoulos, D. S., 2016, "Challenges and opportunities in edge computing", IEEE International Conference on Smart Cloud (SmartCloud), pp. 20-26

[8] Yuan, J., Li, X., 2018, "A Multi-Source Feedback Based Trust Calculation Mechanism for Edge Computing", IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 819-824

[9] Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L., 2016, "Edge computing: Vision and challenges", IEEE Internet of Things Journal, 3(5), pg. 637-646

[10] JoSEP, A. D., KAtz, R., KonWinSKi, A., Gunho, L. E. E., PAttERSon, D., RABKin, A., 2010, "A view of cloud computing", Communications of the ACM, 53(4)

[11] Satyanarayanan, M., 2017, “The emergence of edge computing”, IEEE Computer Society, 50(1), pg. 30-39

[12] 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

[13] Zhu, J., Chan, D. S., Prabhu, M. S., Natarajan, P., Hu, H., Bonomi, F., 2013, “Improving web sites performance using edge servers in fog computing architecture”, IEEE Seventh International Symposium on Service-Oriented System Engineering, pp. 320-323

[14] Beloglazov, A., Abawajy, J., Buyya, R., 2012, “Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing”, Future generation computer systems, 28(5), pp. 755-768

[15] Houmansadr, A., Zonouz, S. A., Berthier, R., 2011, “A cloud-based intrusion detection and response system for mobile phones”, IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W), pp. 31-32

[16] Yu, W., Liang, F., He, X., Hatcher, W. G., Lu, C., Lin, J., Yang, X., 2017, “A survey on the edge computing for the Internet of Things”, IEEE access, 6, 6900-6919

[17] Garg, S. K., Versteeg, S., Buyya, R., 2013, “A framework for ranking of cloud computing services”, Future Generation Computer Systems, 29(4), pp. 1012-1023

[18] Beck, M. T., Maier, M., 2014, “Mobile edge computing: Challenges for future virtual network embedding algorithms”, The Eighth International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP), pp. 3

[19] K. Zhang, Y. Mao, S. Leng, Y. He, Y. Zhang, 2017, "Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading", IEEE Veh. Technol. Mag., vol. 12, no. 2, pp. 36-44

[20] C. Regazzoni, A. Cavallaro, Y. Wu, J. Konrad, A. Hampapur, 2010, "Video analytics for surveillance: Theory and practice", IEEE Signal Process. Mag., vol. 27, no. 5, pp. 16-17

[21] Y. Mao, J. Zhang, Z. Chen, K. B. Letaief, 2016, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices", IEEE J. Sel. Areas Commun., vol. 34, no. 12, pp. 3590-3605

[22] Alsenani, Y., Crosby, G., & Velasco, T., 2018, “Sara: A stochastic model to estimate reliability of edge resources in volunteer cloud”, IEEE International Conference on Edge Computing (EDGE), pp. 121-124

[23] Jøsang, A., Ismail, R., & Boyd, C., 2007, “A survey of trust and reputation systems for online service provision”, Decision support systems, 43(2), pp. 618-644

[24] Pinto, S., Gomes, T., Pereira, J., Cabral, J., Tavares, A., 2017, “IIoTEED: an enhanced, trusted execution environment for industrial IoT edge devices”, IEEE Internet Computing, 21(1), pp. 40-47

[25] Rehiman, K. R., Veni, S., 2017, “A trust management model for sensor enabled mobile devices in IoT”, International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC), pp. 807-810

[26] Alshehri, M. D., Hussain, F. K., 2017, “A centralized trust management mechanism for the internet of things (CTM-IoT)”, In International Conference on Broadband and Wireless Computing, Communication and Applications, pp. 533-543

[27] Kim, E., Keum, C., 2017, “Trustworthy gateway system providing IoT trust domain of smart home”, IEEE Ninth International Conference on Ubiquitous and Future Networks (ICUFN), pp. 551-553

[28] Asiri, S., Miri, A., 2016, “An IoT trust and reputation model based on recommender systems”, 14th Annual Conference on Privacy, Security and Trust (PST), pp. 561-568

[29] Mendoza, C. V., & Kleinschmidt, J. H., 2016, “Defense for selective attacks in the IoT with a distributed trust management scheme”, IEEE International Symposium on Consumer Electronics (ISCE), pp. 53-54

[30] Echeverría, S., Klinedinst, D., Williams, K., Lewis, G. A., 2016, “Establishing trusted identities in disconnected edge environments”, IEEE/ACM Symposium on Edge Computing (SEC), pp. 51-63

[31] Sharma, A., Pilli, E. S., Mazumdar, A. P., Govil, M. C., 2016, “A framework to manage trust in internet of things”, International Conference on Emerging Trends in Communication Technologies (ETCT), pp. 1-5

[32] Wang, J. P., Bin, S., Yu, Y., Niu, X. X., 2013, “Distributed trust management mechanism for the internet of things”, Trans Tech Publications in Applied Mechanics and Materials, Vol. 347, pp. 2463-2467

[33] Kagal, L., Finin, T., Cost, R. S., & Peng, Y., 2001, “A framework for distributed trust management”, In Second Workshop on Norms and Institutions in multi-agent systems.

[34] Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Incremental singular value decomposition algorithms for highly scalable recommender systems. In: Proceedings of the 5th International Conference in Computers and Information Technology (2002)34 Francesco Ricci, Lior Rokach and Bracha Shapira

[35] Risawandi & Rahim, Robbi. (2016). Study of the Simple Multi-Attribute Rating Technique For Decision Support. International Journal of Scientific Research in Science and Technology. 2. 491-494.

[36] A. Paterek, "Improving Regularized Singular Value Decomposition for Collaborative Filtering," Proc. KDD Cup and Workshop, ACM Press, 2007, pp. 39-42.

[37] Oktavianti, E & Komala, N & Nugrahani, F. (2019). Simple multi attribute rating technique (SMART) method on employee promotions. Journal of Physics: Conference Series. 1193. 012028. 10.1088/1742-6596/1193/1/012028.

[38] Brand, M. (2002). Incremental singular value decomposition of uncertain data with missing values. In A. Heyden, G. Sparr, M. Nielsen & P. Johansen (Eds.), LNCS : Vol. 2350. Proceedings of the seventh European conference on computer vision (pp. 707–720). Berlin: Springer.