Architecture and Framework for Reliability of Edge-assisted IoT Systems

A DISSERTATION

Submitted in the partial fulfilment of the

requirements for the award of the degree

of

MASTER OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

by

Akriti Sood

17535001

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY ROORKEE

ROORKEE - 247 667 (INDIA)

May, 2019

CANDIDATE’S DECLARATION

I hereby declare that the work which is being presented in the dissertation entitled **“ Architecture and Framework for Reliability of Edge-assisted IoT Systems”** towards the partial fulfillment of the requirements for the award of the degree of **Master of Technology in Computer Science and Engineering** submitted in the Department of Computer Science and Engineering, Indian Institute of Technology Roorkee, Uttarakhand (India) is an authentic record of my own work carried out during the period from July 2018 to May 2019 under the guidance of **Dr. Sateesh K. Peddoju**, Associate Professor, Department of Computer Science and Engineering, IIT Roorkee.

The matter presented in this dissertation has not been submitted by me for the award of any other degree of this or any other Institute.

Date: (**Akriti Sood**)

Place: Roorkee **Enroll. No -17535001**

CERTIFICATE

This is to certify that the above statement made by the candidate is correct to the best of my knowledge and belief.

Date: **(Dr. Sateesh K. Peddoju)**

Place: Roorkee Dissertation Supervisor

PUBLICATION

Part of the work submitted in this dissertation has been communicated to the publication in the following conference:

Akriti Sood, Sateesh K. Peddoju, ”Reliable IoT Systems using Edge Computing”, *The 16th IEEE International Conference on Mobile Ad-Hoc and Smart Systems(MASS)*, CA, USA, November 2019.

ACKNOWLEDGEMENTS

I, Akriti Sood, would never have been able to complete my dissertation without the guidance of my supervisor, help from friends, and support from my family and loved ones.

First and foremost, I would like to extend my heartfelt gratitude to my guide and mentor Dr. Sateesh K. Peddoju, Associate Professor, Department of Computer Science and Engineering, Indian Institute of Technology Roorkee, for his invaluable guidance, and encouragement and for sharing his broad knowledge. His wisdom, knowledge and commitment to the highest standards inspired and motivated me. I would also like to thank Mr. Jitender Grover, Research Associate, for his precious time and guidance all through the thesis project. I thank Dr. Debi Prasad Kanungo, Professor of CBRI, Dr. Rajib Panigrahi, Associate Professor of ECE who helped us to utilize the facilities and datasets of their NMHS research project(NMHS/2017-2018/MG47/31).

I am grateful to B.Tech team working under NMHS project for their Landslide prediction machine learning model which I have used in my thesis work.

I would like to thank my fellow classmate, Meenakshi Raikwar, M.Tech student, for her assistance and help during the project. I am grateful to members of HPC lab who assisted in my thesis work directly or indirectly.

I am also grateful to the Dept. of Computer Science and Engineering, IIT-Roorkee for providing valuable resources to aid my research. Finally, hearty thanks to my parents and siblings, who encouraged me in good times, and motivated me in the bad times, without which this dissertation would not have been possible.

Dedication

To my Family, for their Love and Support

ABSTRACT

Internet of Things (IoT) is the interconnection of Internet-enabled devices that can share information with each other or to Cloud servers. The communication between the delivering nodes and the Cloud server is vulnerable to link failures. In real time scenarios, this connection loss might hinder the decision making at the Cloud due to non-reception of needed data at the right time. Eventually, affecting the reliability of the IoT systems. Edge servers, placed near to the IoT node on the edge of the core network, will have limited computing and storage resources that can address the reliability of the systems. They will collect data from the IoT node and forward data to the Cloud. In this way even if the connection to the Cloud is lost, the Edge server would still be able to process and analyze sensed data to produce meaningful results similar to Cloud. Yet another challenge is the failure of links between IoT nodes and the Edge servers. This research work is aimed at making the data processing and alarm generation as reliable as possible so that even if the connection is lost between the IoT node and Edge server or Cloud server, the data could still be processed and results are obtained. The work in this thesis is expected to produce a reliable data processing system which can be used for real-time response. The proposed mechanism has been implemented on the laboratory setup to prove that, though there might occur transmission and processing delays during switching of the system, after all, is reliable. The proposed model is tested for Landslide Early Warning System (LEWS) where connection failure between the IoT node and Cloud can be catastrophic and cause danger to human lives.

Contents

List of Figures

List of Tables

1 Introduction

In the IoT paradigm, many of the objects around us are on the network in one form or another. IoT is the connection of internet enabled devices which generate huge data volumes. This data needs to be stored, processed and presented in an efficient and fairly understandable form [1]. Cloud computing provides the infrastructure for shared pool of online resources which can be accessed from anywhere,anytime. IoT-Cloud integration is surely one among many exceptional technologies trending in today's world. Various facilities and products like Smart homes, Smart City, eHealth etc are trending for well-being, luxury and welfare of public. These are the result of in- tegration of IoT and Cloud. IoT is expanding its reach in different domains like Banking, finance, Agriculture, Entertainment, disaster management and what not. While some of them are merely for the recreation and comfort of public, other systems are designed to have life saving applications.

1.1 Motivation

In applications like Health monitoring, Autonomous vehicles where the sensors generate huge data volumes which need to be processed immediately are very sensitive to End-to-End delay. In Health monitoring, if there is any delay in processing body sensors' data or the server is not available for a long time then the patient's health condition would not be studied. It can have adverse impact on patient's life in case of emergency. Similarly, in the application of autonomous vehicles, the delay in processing the data of sensors embedded in the vehicle, which control the vehicle to move in right direction can result in serious accidents. Hence, the Cloud's availability is essential at all the time in these type of real-time systems.These kind of systems are very sensitive to the transmission and processing delay. The application must be tolerant to any kind of unforeseen events like failure of underlying services as well as faults that are related to internet infrastructure, broken communication links or physical hardware failure when dealing with realtime services[2]. Hence, there is a need to develop a reliable infrastructure that operates normally despite failure at cloud level. One of the ways to achieve fault tolerance would be to use redundancy in the system. Typical methods to introduce redundancy can be duplicating servers, controllers, I/O, and networks[2]. A distributed fault tolerant algorithm, [3] is designed that is run on all redundant controllers asynchronously. This is done to maintain reliability in the system.

1.2 Problem Statement

In the wake of importance of Reliability and Fault tolerance in real world applications, a system is proposed here to enhance the reliability of the IoT-Cloud system. The reliability of data processing should be ensured at every level of hierarchy in the system. One such application where fault tolerance and reliability are the key concerns is Landslide Early Warning System(LEWS). LEWS is a system which records and analyzes the landslide's changing characteristics and is able to predict the occurrence of landslides with various warning levels in real-time. Landslides cause huge destruction to the mankind, landscapes and ecosystem. Although, it is very difficult to prevent a natural hazard but we can surely take some measures to reduce the harm and misery caused by them. If the administration is alerted sometime prior to the landslide then some plan of action can be followed to evacuate people from the area. In this way, although the area might be destroyed but human lives could be saved. The sensors deployed on the landslide, determining the characteristics of the slope and soil are continuously sending data to the Cloud where rigorous machine learning algorithms are running to predict the event of landslide. In the event of non availability of computing resources, the system might not predict accurately and the system would fail. So here, we need to have computing resources up and running all the time.

1.3 Objective

The objective of this thesis work can be briefly summarised with following points :

- The foremost motive is to provide a back-end device working instead of Cloud in case of any situation of Cloud failure.

- The second objective is to maintain reliability even at the intermediate layer of the architecture. If one of the nodes is not reachable then information should be diverted to some other node at the intermediate level and hence to the cloud.
- The designed system is to be checked and tested against LEWS. The system would predict the occurrence of landslide in a certain region.

1.4 Contribution

For making the IoT system reliable to overcome possible failures in the network and Cloud, the proposed solution employs Edge computing in the system. Edge Computing is nothing but bringing the store and compute resources closer to the edge of the IoT node. Due to long distance and wireless connection between IoT node and Cloud, the connection is vulnerable to failures. The objective here is to introduce an edge server at the edge of IoT node which would work as a fallback service to mask the failure. Data processing could still be carried out on the edge servers and appropriate responses are generated until the connection is regained.

The data for a particular region would be forwarded to edge server. In case the edge node for a region is not active, an arrangement is made to maintain the reliability at the edge level by switching connection from one edge node to another edge node. Hence, one edge node would be able to receive data from more than one region. The model is tested with the laboratory setup and results and analysis is done.

1.5 Organization of Dissertation Report

Rest of the dissertation is organized as :

In Chapter 2, we discuss about the background of the dissertation work. We discussed about the IoT and Cloud applications and integration. Introduction to Edge computing and its promising features are discussed. Also, we talk about Landslides and Landslide Early Warning System(LEWS) application. We discussed about the related work done by other authors and discuss their contributions. Also, we discussed about the research gaps based on which we proposed the model to overcome these research challenges

In Chapter 3, the architecture design of proposed system is illustrated. Each module of the design is further discussed. Also, we proposed the model for reliable IoT-Cloud system. Various scenarios have been addressed separately giving a more clear understanding.

In Chapter 4, we presented the Experimental setup for testing the proposed design and discussed different tools used for the test bed. We discussed about algorithm designed for LEWS. We presented the results based on the experiment performed based on various parameters.

In Chapter 5, we concluded the dissertation report and the future work.

2

Literature Work

IOT is a field of computer science and networking in which there is a network of smart devices which share and exchange information with one another. This information sharing can happen through any communication technology like WIFI, Blue-tooth etc. The market of IoT is getting larger day by day. Sometimes, it is as playful as a child's toy and might be as serious as the wearable devices for security of masses. More than the humans, things are connecting nowadays. There was an estimate that around 8.4 million IoT devices were in use in 2017. This technology has been in use in almost all area of life like -Smart Home, Smart City, Smart Agriculture, Wearables for safety, Smart grid, Smart In- frastructure, Smart Poultry, Smart Transportation and what not. IoT allows people and business to be more connected to the world which help them do some meaningful high-level work. IoT might be the network of various sensors which can sense anything from biological parameters to motility parameters. For smooth functioning, control and merging of various heterogeneous inputs,micro-controllers are used to which outputs of various sensors are fed. Micro-controllers are coded to perform desired function.

2.1 Background

2.1.1 IoT and Cloud

Huge amount of data generated by these IoT devices need to be stored, maintained and processed for analysis and future reference. So,maintaining this huge data in in-house system is'nt practical anymore. The cost of maintaining, upgrading the system is too high which might not be afforded by some small companies. Hence, commercially as well as effectively it is beneficial to use Cloud for Storage and processing. Various cloud providers provide built in IoT services like Amazon web services, Google Cloud, IBM Watson IOT platform, Oracle, Salesforce, Bosch, Azure IoT suite, Cisco IoT Cloud Connect, SAP and many more.

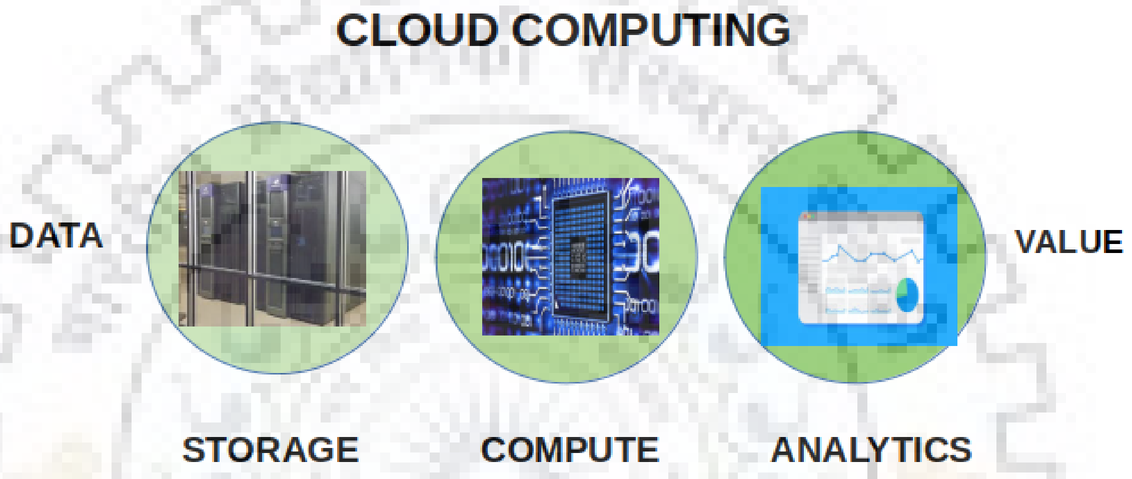


FIGURE 2.1: Cloud Computing

The cloud provider should be chosen according to our business requirement. Cloud provides good performance, high productivity, excellent speed,cost reduction, and reliability. There are mainly 3 types of cloud services :
1. Private Cloud Service, which run on organi- zation's own infrastructure and is not available to the general public.

2. Public Cloud Services provide cloud to multiple users through internet.

3. Hybrid Clouds combine public and private clouds, bound together by technology that allows data and applications to be shared between them.

Mostly,cloud services fall into three major categories : Infrastructure as a service (Iaas) : In this kind of service,we can rent IT infrastructure- servers and virtual machines (VMs), storage, networks, operating systems from a cloud provider on a pay-as-you-go basis. Platform as a service (Paas) : PaaS refers to on-demand environment for developing, testing, delivering and managing software applications. It makes it simpler for the developers to quickly create web or mobile apps, without having to worry about setting up or managing the underlying infrastructure. Software as a service (Saas) : SaaS is a method for delivering software applications over the Internet, on demand and typically on a subscription basis. Cloud providers host the software applications and infrastructure. In our study, we exploit the cloud services in order to run various data analysis applications and store large data gathered from various sensors. Data gathered from the sensors is taken by micro controllers which export the data to the clouds using various communication technologies like, Wifi, Ethernet.

2.1.2 Edge Computing

Pushing intelligence to the edge of the network is generally called as Edge computing. Back in the old days, all the main computational power was done by the main power and the clients were just a dumb screen. Then came client-server architecture where some intelligence was added to the client but still, most computing power was there in the main server. In the modern times, where we have the world of Internet of Things, there are thousands of millions of billions of devices. If we allow all these devices to send all of their data to main server for computation, then there might be a big problem. There may not be enough bandwidth for these devices to constantly communicate with the server. The servers themselves would get overloaded with immense data. So, a number of years ago there was this idea of pushing intelligence to the edge, so instead of sending all the data to the main server, why don't these edge devices process the data partially and send just the result to the server. For instance, Surveillance Cameras are the perfect target for edge computing. Earlier, these cameras were dumb, they just constantly sent data back to the DVR main video recorder server. So, 24*7, there was video streaming on the server and it used to decide what to do with the video. In the old days, this architecture was fine because cameras were less in number but now, as we have surveillance cameras everywhere, it is a overhead to stream video to the server 24*7. Connections are very inexpensive and there would be gigabytes of data transmission per second. To the top of that, if these cameras send high-definition videos then it would completely saturate the bandwidth. Hence, the idea was introduced to add some processing power to the cameras so that they only send data when they detect some motion. There are various other applications which showcase substantial improvement when introduced with Edge computing.

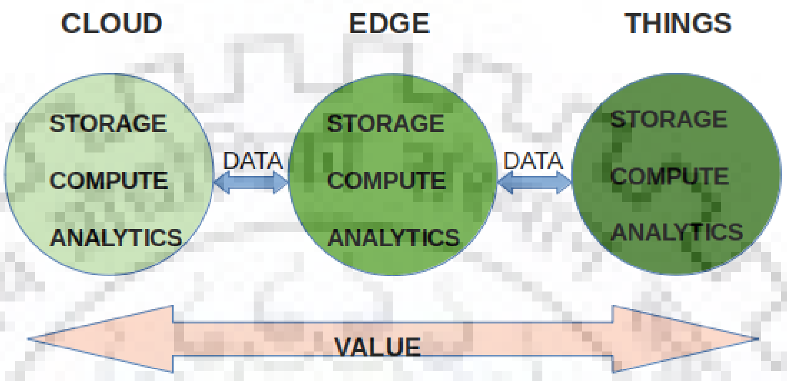


FIGURE 2.2: Edge Computing

2.1.3 LEWS

Landslides are one of the most dangerous and destructive natural hazards that cause significant damage to economic objects and human lives. The movement rate of landslides varies from the slow movement of material in millimetres or centimetres range per year to a sudden avalanche of a large quantity of debris. There are many examples of the negative manifestations of landslides in various regions of India. Listing from the year 2000, Amboori lanslide – Kerala, Kedarnath landslide – Uttarakhand, Malin landslide – Maharashtra are among the most destructive ones where thousands of people lost their life and some also got missing. Latest being the landslide in chamoli district of Uttarakhand causing huge discomfort and lose. Uttarakhand state has 24 areas spread in 344 locations including Bhagirathi valley and Kedarnath township which fall under very unstable zones. Here, recurrent landslide problem has been witnessed over the years. There are mainly two causes for a disaster – Natural(inevitable) causes which are beyond human control and order. Human triggered landslides caused by increasing pressure on the environment in the form of construction work, unlawful mining, heavy machine vibrations in ground and hill-cutting. All these mentioned activities can cause unstability of the soil and slope surface which is the reason behind slope movement and displacement. Earthquakes, heavy rainfall, ground water dis-balance, soil erosion are few factors which effect the stability of slope naturally. Landslides might often lead to another natural calamities like floods to further worsen the living conditions. One of the main causes of emergencies and negative manifestations of landslides is the lack of a unified monitoring system for the prediction and early warning of the landslide. Landslide monitoring is generally not practiced in India and an effective warning system is required to help the people get evacuated in such conditions.

Keeping in view the societal and strategic relevance of landslide disasters, there is a need to deploy monitoring and warning system in such areas. Several technologies and systems have already been proposed and designed for monitoring landslides. The prevalent systems differ in the technologies or protocols used for data streaming, analysis algorithms, offline-online prediction, wireless technologies used for data transfer, operating cost involved and various other factors. LEWS is the Warning System designed for Landslides which monitors the prone locations, collects the data related to soil and rainfall characteristics, processes that data into meaningful information, applies machine learning algorithm for rigorous analysis of the data to predict the occurrence of landslides beforehand. This will help the respective State Disaster Management authorities in the landslide risk reduction. It will help the administration to prepare emergency services in advance. In few systems, there are no prediction algorithms running. Warnings are just issued on the basis of threshold level on the basis of IF-ELSE paradigm. In the system designed by Chaturvedi, P. et al. [4], the field sensors used at the Tangni Landslide site are in-place inclinometers (IPI), vibrating wire piezometers, tipping bucket rain gauge and extensometers. The data is transmitted automatically using GSM Modems, stored, processed and visualized on a DTRL base-station computer. IPIs consist of string of MEMS sensors installed in casing within a borehole. Multiple IPI's are linked together by Gauge tubes. Each IPI is connected to the data logger which is power source of sensors. In some systems, manual expertise and intervention is also there which does not make the system fully automatic. Also, none of the systems has used Edge computing for improving the latency and increasing the reliability of the system.

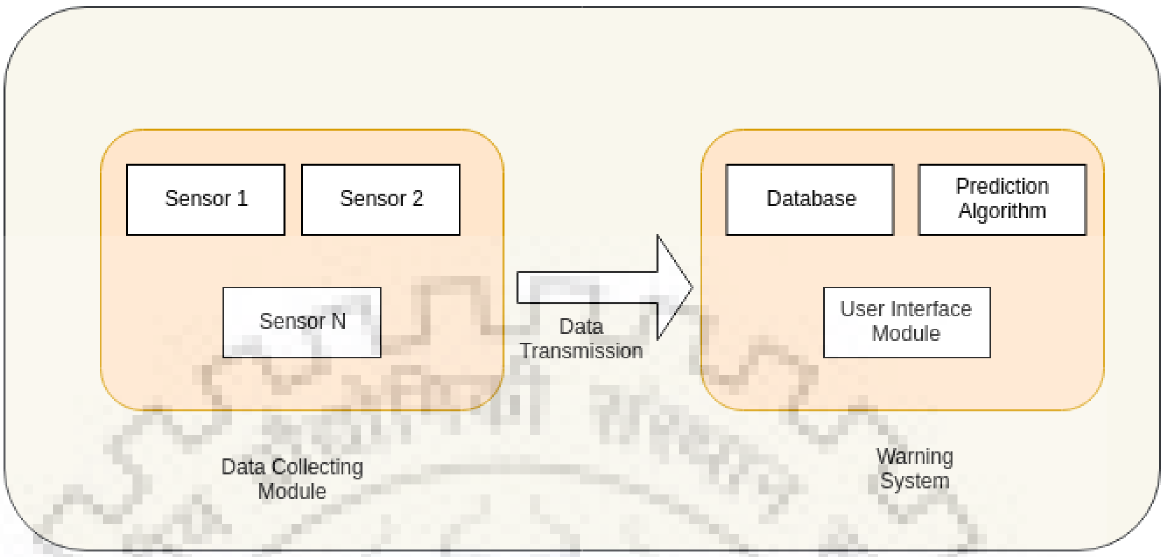


FIGURE 2.3: Design for LEWS

2.2 Related Work

A lot of research and work has been going on in the field of Edge and Cloud. This section talks about the previous work that has been done in the Edge and Cloud areas. The motive is to prove and ensure that Edge computing adds to the various advantages provided by the cloud. IoT-Edge-Cloud integration has been studied and worked upon by the researchers and technologists in various applications. Paul et al. [6] talks about Edge computing in Healthcare monitoring system. Due to privacy and security concerns, the hospitals would not like to have their patient's data stored outside and hence, in this domain simple sensor-to-cloud architecture is not viable. The authors also emphasize that there can always be the bleak case of a network failure or data center failure which can put their patient's health at risk. Hence, they propose to extend the centralized cloud computing architecture to a distributed one. This distributed architecture serves the purpose for both security and overhead time taken for data to ho back and forth from the cloud to devices.

Cao et al. [7] have proposed a system to detect the fall in real time. They have named it as U-Fall. The system distributes the analytic throughout the network by dividing the detection task between the edge devices (e.g., smart phones attached to the user) and the server (e.g., servers in the cloud). They have devised new algorithms for front-end run and back-end run. They put forward that such system designs not only provides low response time but also consumes as low battery power as possible on smartphone. Edge computing can not only be used with static networks but also can be exploited in mobile devices. Liu et al. [8] employs edge computing in the field of Augmented Reality(AR). AR requires speedy and accurate detection and understanding of physical ecosystem so that virtual contents can synch with the real world. The image processing algorithms like SSD,YOLO and Faster R-CNN are too complex to be run on the mobile devices to detect the objects. Neither can a mobile device perform any advanced computer vision the edge-based MAR system., authors have desinged a DARE (dynamic adaptive AR over the edge) protocol that enables the edge-based MAR system to dynamically change AR configurations and computation resource allocations according to wireless channel conditions and available computation resources on the edge server. Via this adaption, the DARE protocol trades off the quality of augmentation (QoA) against the service latency.

Zamora-Izquierdo et al.[9] design a three-tier open source software platform at local, edge and cloud planes. At the local plane, Cyber-Physical Systems (CPS) interact with crop devices to collect data and perform real-time atomic control actions. The edge layer of the platform is in charge of monitoring and managing main tasks near the access network to increase reliability of the system against network failures. The cloud platform gathers live and historic records and conducts data analytics modules in a FIWARE deploy-Edge Computing. Distribution of application in hybrid Edge-Cloud architecture can be performed in number of ways. Ashouri [10] studies the various parameters on the basis of which application should be distributed between edge and Cloud. These parameters are Response time ,accuracy, availablity, energy consumption, security, privacy, cost, elasticity, scalabilty.

The network disconnectivity in highly dynamic and volatile environments has been discussed in [11]. In these type of environment, the architecture should be highly resilient to the failure caused by network disconnections. The authors have presented a novel edge computing system architecture that delivers failure-resistant and efficient applications by dynamically adapting to handle failures. All computing nodes, including the edge server and the available mobile devices, are connected using peer-to-peer communication interfaces, the main and the backup ones. When the edge server is inaccessible, the backup mechanism makes the mobile devices to interact with each other directly, with a cluster of mobile devices providing the edge services. The authors [12] talk about the ultra-reliable low latency communications (uRLLC) in the fifth generation mobile communication system. The tradeoff between the latency and reliability in task offloading to MEC is studied. The task is partitioned into sub-tasks and offloaded to the multiple edge nodes. Three algorithms namely, based on heuristic search, reformulation linearization technique and semi-definite relaxation, respectively are designed to to solve the problem of optimizing edge node candidates selection, offloading ordering and task allocation.

IoT-Edge-Cloud combination is highly useful in autonomous vehicles (AVs) in smart cities. The integration can be used to enhance the reliability of the Vehicle-to-infrastructure network. A novel algorithm [13] based on Matching Theory is proposed to jointly optimize AVs-to-SBSs(small base stations) association and bandwidth allocation to maximize the reliability of the Vehicle-to-infrastructure network.Because of varying task types and random wireless channel variations, it is very important to optimize the end-to end quality-of-service in V2I networks. The proposed framework uses tools from Labor Matching Markets to effectively perform distributed association of AVs to SBSs, while considering the latency needs of AVs along with the limited bandwidth and computational resources of SBSs. The proposed algorithm is proved to converge to a core allocation between AVs and SBSs, thus ensuring the stability of the V2I network when using distributed implementations. Javed et al. [14] propose a fault-tolerant architecture, CEFIoT for Edge and Cloud which solves the purpose of having similar software stack at both the edge and cloud for managing and moving data processing between edge and cloud. CEFIoT framework is composed of small cluster of embedded Linux devices together with Kubernetes and Apache Kafka to implement the fault tolerance capabilities. The Kafka publish/subscribe (pub/sub) platform is used as unified high performance data replication solution for both edge and cloud.

2.3 Research Gaps

The research work proposed above talk about the enhanced efficiency of the system in terms of reduced latency and ultra reliability by introducing Edge Computing. They explained how Edge can work as a backup when the Cloud server is unaccessible. The system should be able to survive failure at any layer of architecture. People have proposed Edge Computing in controlled environment. Implementing Edge Computing in remote areas, rough terrains, hilly areas is something not explored or talked about till now. The proposed system introduces Edge technology to IoT-Cloud infrastructure in remote areas to construct a reliable system which can sustain Cloud failure. This proposed solution also aims at reliability at the Edge level which means that it also has the mechanism to switch to another Edge server when one Edge fails. Edge Computing has not been explored in the area of Landslide detection. So, the proposed system can be extended to be part of the LEWS and maintain the reliability of the application.

2.4 Summary

This Chapter of the thesis report is focused on two aspects. Firstly, I have talked about the background technologies, IoT and Edge Computing, that are the core part of the proposed model. Here, the application, LEWS, against which the model would be tested is also explained. We have talked about Landslides in general and what LEWS is. Secondly, this section talks about the previous work done in the field of edge computing and reliable IoT systems. It is focussed on the various use cases where edge computing has been used to address the issues like security, privacy, latency sensitivity, reliability etc. Further, the discovered research gap is also explained in section 2.3.

3 Proposed Model

3.1 Architecture of the Proposed System

In order to maintain the reliability of the IoT system, the concept of Edge computing has been introduced in the framework. The Edge server would work as a complement to the cloud server and would provide uninterrupted service. The system architecture is composed of following three layers :

1. Data collection layer
2. Edge Server
3. Cloud Server

1. Data Collection Layer

This section majorly deals with the hardware. In this particular section of the proposed system, data gathering devices are being used. These devices can be sensors, Cluster head or any other electronic equipment. A device which produces any output by observing the changes in quantities or events can be defined as a sensor. Depending on the Application to be designed, the desired sensors are chosen. Adhering to the understanding of the project, the number of required devices and their placement in the area is evaluated. The devices might be or might not be Internet enabled. Since, we are designing an IoT system here, so the devices are capable of sending collected data via the internet to another end. The frequency at which these sensors should record and supply the data can be set. This network of sensors is called as Wireless Sensor Network(WSN). There are various protocols and message passing algorithms for transferring data from one end to another in WSN. A suitable technology should be chosen according to the features of application like Transmission delay, security, reliabilty etc.

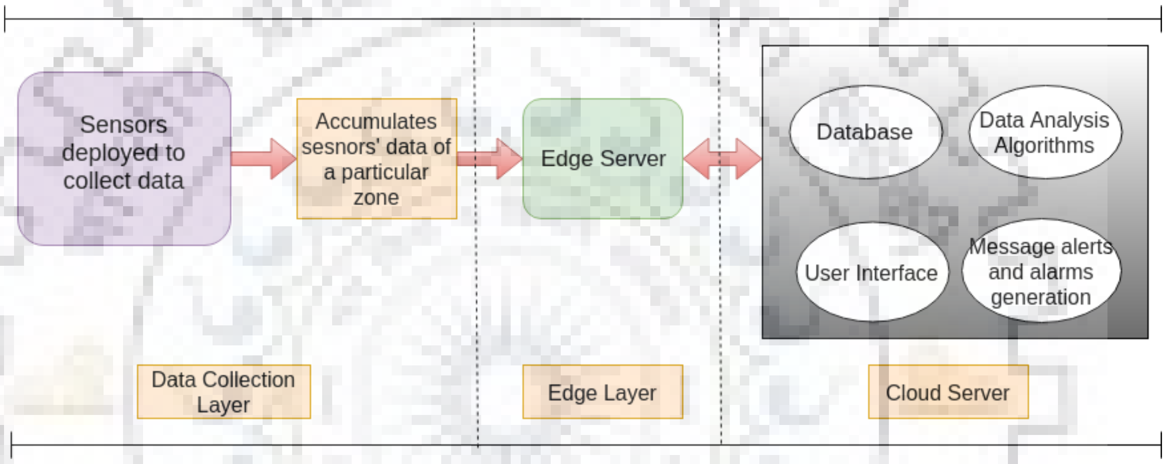

Sensors deployed to collect data
Accumulates sesnors' data of a particular zone
Edge Server
Database
Data Analysis Algorithms
User Interface
Message alerts and alarms generation
Data Collection Layer
Edge Layer
Cloud Server

FIGURE 3.1: Architecture Design of the system

2. Edge Server

The data consumed from the sensors has to be stored and processed at a safe location. Cloud is a shared pool of resources and services which are leveraged in a very feasible and manageable manner. Cloud may be at a far off place from the data sources . In real time scenarios, end-to-end latency and physical proximity play a key role in the efficiency of the system. It is more functional and effective to place the computing device in close proximity to the sensors, end users and the site under study. This lead to the emergence of edge Computing which is nothing but providing the services as close as possible to the edge of the network. Edge server leverages the infrastructure of cloud computing by placing the computing resources(cloudlets) at the Internet edge. The cloudlets have limited processing and storage resources. Edge Server can be deployed at any device, for instance, mobile of the user, cell tower, vehicle, laptops etc. Cloudlets have a limited processing power which can be harnessed according to the application. The Edge Server collects data from the Data Collection layer, and forwards it to the main server. Hence, it acts as an intermediate node between first and third layer.

3. Cloud Server

The Cloud Server is the remote powerful machine, which maintains the database, regulates data analysis algorithms and provides user interface. Server collects and accumulates sensor data in the database. It receives data values from the edge server. Cloud can communicate with the Edge via various protocols. Server has a substantial memory and pool of resources. It runs efficient algorithms for analyzing the data, find hidden patterns and generates the results.

3.2 Proposed Model

The proposed system enrolls Edge Computing paradigm in its architecture to support the reliability requirement of the IoT-Cloud system. The sensors' data would be transmitted to Cloud server from the IoT node via an Edge Server in the middle. In the proposed system, we would be dealing with different hardware and communication protocols. The choice of sensors would be according to the application under development. In this thesis work, application considered is Landslide Prediction. The IoT node could be any device enabled with Internet like an IoT sensor, cluster head, Arduino module, Raspberry pi,, PC, mobile etc. The edge node can be realized as a simple Raspberry Pi module or a full fledged PC. The edge server can be placed at any location near to the IoT source node which is safe and less prone to network and connection issues. There is a generic assumption that the Edge node has enough memory and processing resources to execute small algorithms.

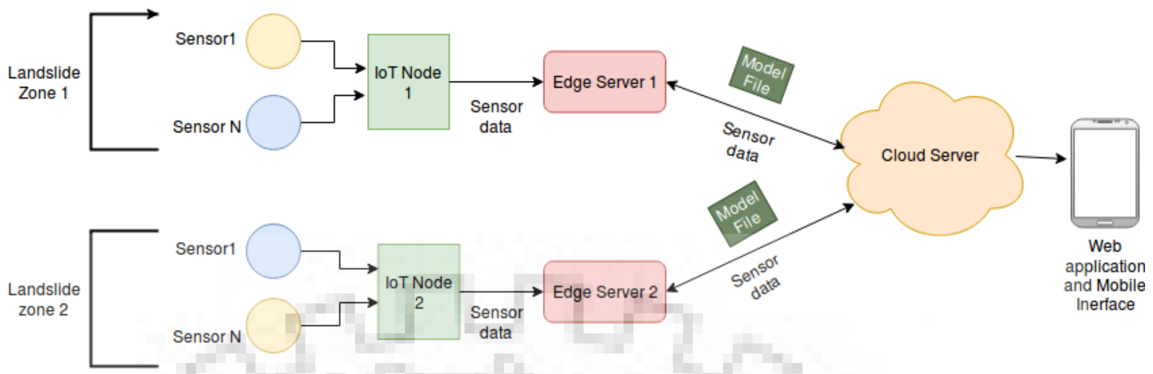


FIGURE 3.2: Proposed Edge-Cloud model for reliability of system

3.3 Methodology of Proposed Solution to various scenarios

3.3.1 Addressing connection loss to Cloud

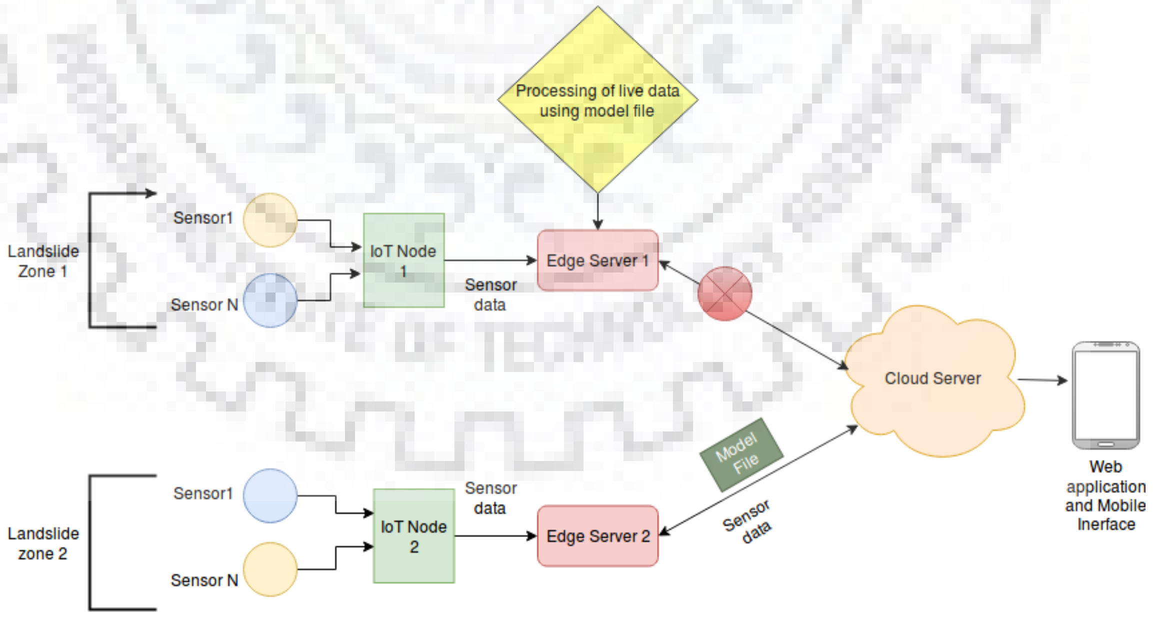


FIGURE 3.3: Scenario when the link to cloud fails or cloud layoffs

When the link between the edge server and Cloud server breaks due to network failure then this edge server works as the proxy and completes the processing needs on Cloud server's behalf. Although Edge server has its storage and compute resources, they are not at par with the resources available at the Cloud. The machine learning or complex algorithms running on the Cloud cannot run on the edge server due to its resource constraints. The Cloud runs the training and testing procedure on the dataset it receives. During the training step of machine learning algorithm at the cloud, a model file would be generated which is basically the trained model.The saving of model is called Serializaion, while restoring the model is called Deserialization. When we need the same trained data in some other device or at some later time, to avoid the wastage of the training time, we can store trained model. Even though we can run small algorithms or tasks on the edge server, we cant perform the training phase here because of its limited database and compute service. Cloud can store data of about 10 years but edge server can only sustain data of few days. Training phase incurs a lot of resource consumption.

The work here is to make the Cloud send that model file back to the edge server periodically, so that edge server can process the incoming sensor data using some basic code with the model file. In this way, whenever the Cloud goes down or the connection breaks, the edge server would utilize the latest model file to process the incoming data from the IoT node. If after processing, the edge finds any anomaly,it would raise the alarm and further actions can be taken. The edge server would be equipped with actuators to generate alarm or responses. In this way, even if we cant access services from Cloud, we would still have our edge server up and running our desired computations.

3.3.2 Addressing connection failure between IoT node and Edge

The next issue to worry about is what would happen when the edge server is itself down or link to the edge gets broken. To ensure the reliability at even edge level, the concept of connection switching is introduced in the system. As illustrated in the proposed architecture, there would be an edge server for each zone(Coordinator node) which will collect the data from the IoT source node. When the IoT node cannot connect to its dedicated edge anymore or the link breaks during the data transmission, connection switching will take place.IoT node would scan for other edge node in its range and attempt to make connection with other available Edge. If successful, the new edge which it is connected to now would then forward the sensor data of this new IoT node along with the data from old IoT node of its own zone. Fig 5.3 illustrates this connection switching. In this manner, the main server would still receive the data of that zone but from another edge server, and hence data loss is prevented. This method illustrates edge switching.

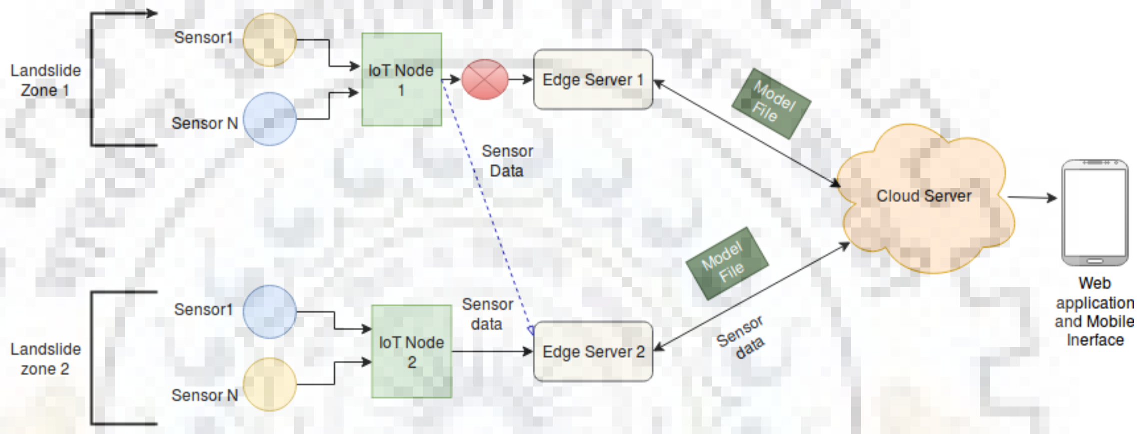


FIGURE 3.4: Connection Switching to other edge server

3.3.3 Addressing Cloud and Edge Failure

In line with the second failure in the system discussed above, there is a possibility that connection to Cloud is also lost, as shown in Fig 3.5. In that case, this second edge works as the proxy server, both for its particular zone and another edge's zone. As this Edge is now receiving data from more than one IoT node, it would test incoming data of both the IoT nodes with the model files of the respective zones. So with this approach, if any edge starts getting data of another zone(another edge) and there is a loss of connection from this edge to cloud then this edge would be analysing the data both for itself and another edge's here only and predict the results for both. As soon as the connection is rebuilt to the cloud, the transmission is resumed.

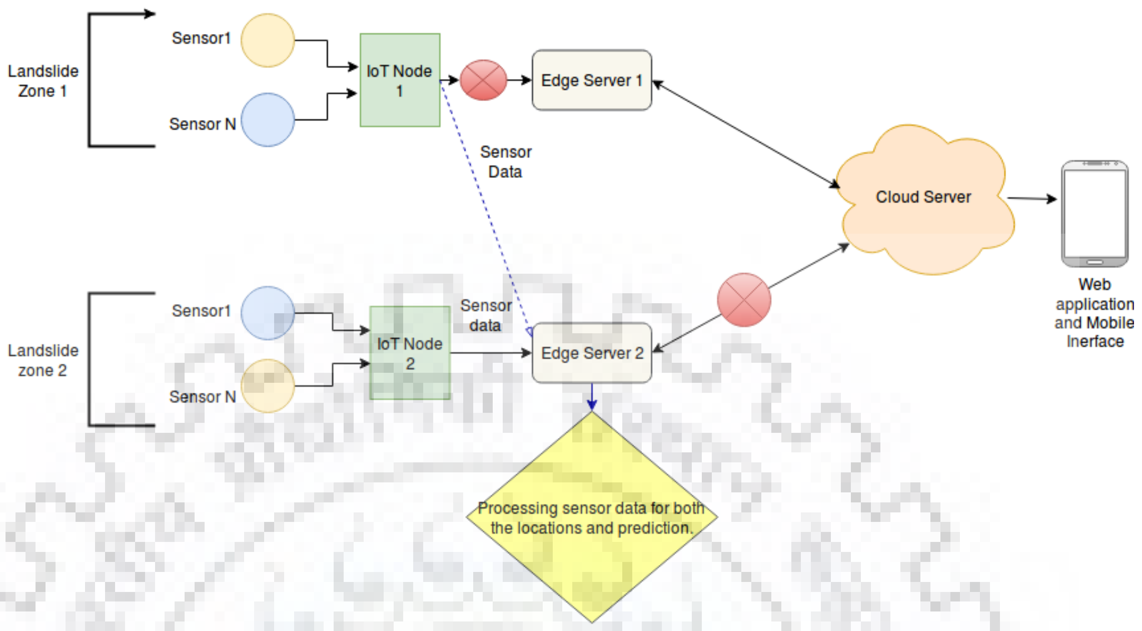


FIGURE 3.5: Edge server processing data for more than one zone

3.4 Summary

This section showcases the generic architecture of the proposed system. The different modules of the model have been talked about and various scenarios have been addressed separately and illustrated with the help of figures.

4 Experiments and Results

4.1 LEWS

For this application, an algorithm is designed which analyzes the rainfall intensity and predicts whether there is a condition for landslide or not. Here the prediction of landslide is determined solely on the basis of rainfall data. The data set has been downloaded from TRMM website which gives the daily mean precipitation values from year 2003 to 2019. Also, the data of 67 landslides from period 2003 to 2019 is downloaded. Rain data is processed and antecedent rainfall analysis mechanism is being used [15]. Antecedent rainfall value of last 30 days is being calculated. This processed rain data is fed to the logistic regression model. If the antecedent rainfall value of last 10 days crosses 80 mm or of last 15 days crosses 180 mm, it would predict the landslide event. So, if the system is working in a conventional manner then the main server would learn from the rainfall data after every fixed time interval and test every data value using machine learning model. But when the Edge machine is not able to connect to main server for any reason, the system does not fail, infact, the edge tests the rain data coming with the model file sent by the server. If it suspects landslide occurrence, then it generates the output accordingly.

To understand the working of the proposed design and to evaluate the results, we developed LEWS. The laboratory setup for testing the proposed system is constructed with the following equipments :

- Coordinator Node : **ESP8266 NodeMCU**

- Edge Server : **Raspberry Pi 3**
- Cloud Server : **Linux Machine**
- Database : **PostgreSQL**
- Messaging protocol for WSN : **MQTT**
- Messaging protocol for Edge-Cloud : **TCP Sockets**

4.2 Experimental Setup

4.2.1 Hardware Setup

1. ESP8266 NodeMCU

ESP8266 is a low cost Wifi chip with a full Tcp/Ip stack. We can upload any program in its MCU to control GPIO pins. This is quiet popular for its small size and low cost in electronic and IoT projects. It works easily with Arduino IDE. It can collect different sensors' values by connecting them to the GPIO board. For our experiment, we are generating random floating point number for the rainfall intensity.

The reason for opting ESP8266 among other available devices is that it is way cheaper than other Arduino modules and other Wifi shields.

FIGURE 4.1: ESP8266(NodeMCU) [16]

2. Raspberry pi

Raspberry Pi is the cheapest and smallest computer out there. It is a credit card sized computer. It has a quad-core arm CPU that does not support all the same instructions as Desktop or laptop x86 CPU. It generally posseses aroung 1 GB of RAM, 4 USB ports, one HDMI port, one ethernet jack, one micro-SD card slot, built in buletooth and Wifi modules. What makes the Pi interesting is 40 GPIO pins that we cannot to evrything from sensors to weather stations to robots. This functionality was built into the pi to teach computer science to relatively young students. It is quiet popular for home projects.

The model used in this project is Raspberry Pi 3. It is loaded with Raspbian(Stretch) Operating system which is very similar to Linux distributions.

FIGURE 4.2: Raspberry Pi [17]

4.2.2 Software Setup

1. MQTT

Message Queuing Telemetry Transfer(MQTT) protocol is a simple messaging protocol, designed for constrained devices with low-bandwidth. Hence, it's the apt solution for Internet of Things applications. MQTT permits us to send commands to control outputs, read and publish data from sensor nodes and much more.

There are four components/concepts of MQTT protocol :

1. The first one is the publish and subscribe system. In such type of system, a device can publish a message on a topic, or it can be subscribed to a particular topic to receive messages.

2. Messages are the data values that you want to exchange between your devices.

3. We can register interest for the messages we want to receive or we can specify where you want to publish the message using Topics.

4. The broker is primarily responsible for receiving all messages, filtering the messages and then publishing the message to all subscribed clients.

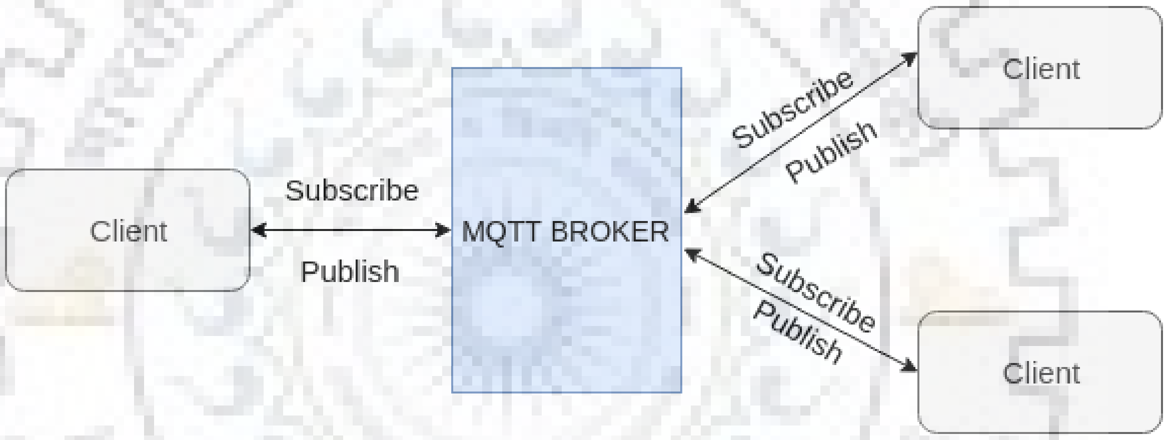


FIGURE 4.3: MQTT Pub-Sub Model

2. TCP Sockets

Tcp sockets are the virtual addresses that identify the network endpoints. These virtual addresses are the combination of Ip address and port number. They are implemented on the Client-Server architecture where client socket is created on the client system and server socket is created on the server system. The server socket waits on a particular port for the client system to make connection request. The client and server can transmit and receive the information using these sockets asynchronously. Sockets allow a single computer to serve many different clients at once, as well as to serve many different types of information. In our system design, client socket program is run on the edge system which is raspberry pi while the server socket is created on the main server. A multi-threaded server program is coded which can accept connections from multiple clients(edges).

3. PostgreSQL

PostgreSQL is a powerful, free and open source object-relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads. PostgreSQL has made its strong hold for its proven architecture, reliability, data integrity, robust feature set, extensibility. PostgreSQL can run on all major operating systems and is ACID-compliant. This database allows us to define your own data types, build out custom functions, even write code from different programming languages without recompiling the database.

4.3 Results

In order to send rain data to raspberry pi, the IoT node, ESP8266 here, first makes wifi connection with raspberry pi. Raspberry Pi has been formed as a wifi access point. It possesses a unique ssid and password which is required to connect to its Wifi. Every nodeMCU has been set with the essentials of the edge server it should connect to. Each edge server also hosts MQTT broker which listens at port 1883. After establishing wifi connection with the raspberry pi, nodeMCU attempts mqtt connection with the broker created at raspberry pi. After this successful connection establishment, nodeMCU starts sharing the rain data at a certain frequency. As soon as raspberry pi gets the data, it sends this rain data to the main server using socket connection. Cloud server runs prediction model for all the edge it is receiving data from.

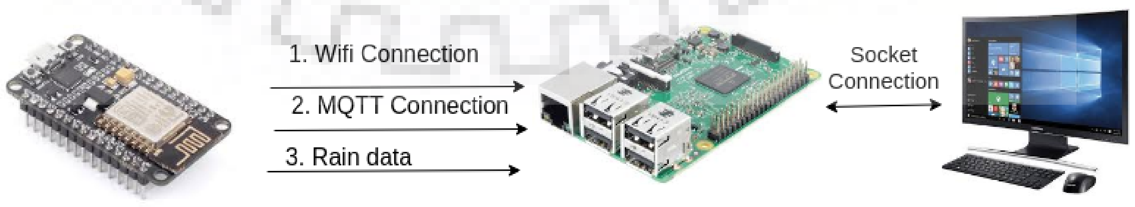


FIGURE 4.4: Communication protocols between 3 level of system

As depicted in fig 3.3, when the nodeMCU detects the connection failure with the raspberry pi, it looks out for other available raspberry pi. It starts scanning the wifi networks in the range and gets connected to the wifi network of nearest raspberry pi. After forming the wifi connection, it forms mqtt connection with the same raspberry pi. As soon as it finishes the re-connection, rain data of this nodeMCU starts getting delivered to the new raspberry pi. So, now this raspberry pi receives rain data from two nodeMCUs. So, it sends rain data of both the nodeMCUs to the main server.

4.3.1 Recovery Time

When the nodeMCU looses connectivity with the raspberry pi, it takes some time to connect to other raspberry pi. This time delay can be called as *Recovery Time*. It subjects to the availability of atleast one working edge server in the region. This switching time includes *time to detect connection loss, time to scan available wifi networks, time to connect to wifi network of raspberry pi and time to make mqtt connection.* It takes approximately 2.5 to make a wifi connection. After many trial runs, it was observed that on average 14 sec are taken to switch connection from one raspberry pi to another.

4.3.2 Comparative Analysis between Edge and Server

4.3.2.1 Prediction Time

To test the rainfall data, prediction algorithm described above is being run on the rainfall intensity values. As the machine learning algorithms consume substantial system resources, depending on the architecture of the underlying system, execution time depends. As the Raspberry Pi has limited resources and power, and has slower processor then the main server, it takes more time to predict the outcome compared to main server. The results are being shown in table 4.2

4.3.2.2 CPU Workload while running prediction algorithm

Running any operation on the computer system involves utilization of CPU. For a certain number of time slices, the cpu is busy, other times it is not. Adhering to the architecture of

TABLE 4.1: Time taken to connect to NodeMCU to Raspberry Pi

Type of Connection	**Average Time taken**
Normal connection establishment from NodeMCU to Pi	2.5 sec
Recovery Time to connect to other Pi	14 sec

TABLE 4.2: Average time to run prediction algorithm

Device	**Average Time taken**
Cloud	1.3sec
Edge	9.1sec

TABLE 4.3: Average CPU workload while running prediction algorithm

Device	**Average CPU percentage in testing**	**Average CPU percentage in training**
Cloud	11%	51%
Edge	34%	N.A

the system, and the application to be run, the percentage of cpu utilization varies. Running machine learning algorithm is highly resources extensive operation. It is observed that it takes around 51% of cpu workload to train and test the LEWS prediction model on the main server. As raspberry pi is not as powerful as a proper linux machine, we cannot run training operation on raspberry pi as it may result in crashing of the system. So, we instead train the model on main server only and just test the data values on the trained model file sent by the main server. Doing this, it just takes on average 34% of cpu utilization to run the model on raspberry pi on a data value. Hence, we saved the cost of running training operation on raspberry pi.

4.3.3 Increase in the CPU Workload on the edge server on increase in the number of connections

For addressing the situation expressed in section 3.2.2, nodeMCU searches for other available raspberry pi in the neighbouring region. It can connect to any of the rpi randomly. In case when a particular raspberry pi starts receiving data of more than one region i.e more than one nodeMCU, it has to process the rain data of all the regions when it suffers from loss to the cloud server. Depending on the number of connections, there would be increase in the CPU workload for the rpi. The graph in Fig 4.5 depicts this increase in the CPU workload against the increase in number of connections upto 3 to a particular edge.

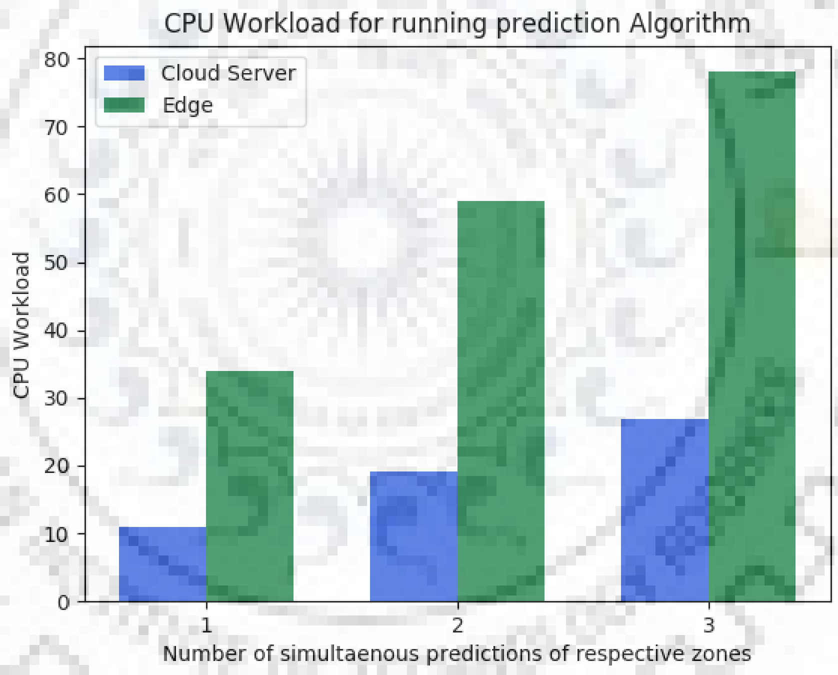


FIGURE 4.5: CPU workload comparison in Cloud and Edge

4.3.4 Impact of frequency at which data is sent from NodeMCU to Rpi

Addressing section 3.2.1, When there occurs connection break from edge to server, prediction algorithm starts running on the edge server on the data coming from nodeMCU. It takes some time to run the prediction algorithm on rpi. As the data is not being stored in the edge server, the data is processed in real time. So, depending upon the frequency at which the data is shooted to the edge server, we observe some missed data packets. Taking the average processing time of 9.1 sec, we observe the following trend of packet miss. So, it can be seen that as we decrease the frequency, there is a decrease in the percentage of missed data packets.

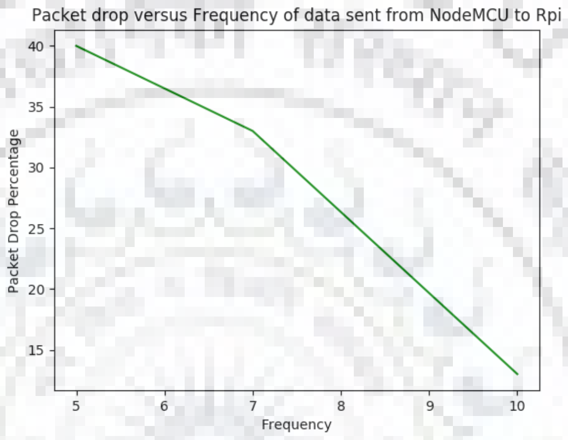


FIGURE 4.6: Packet Drop percentage against Frequency

4.3.5 *Impact of disconnection on the accuracy of the ML model on Cloud server*

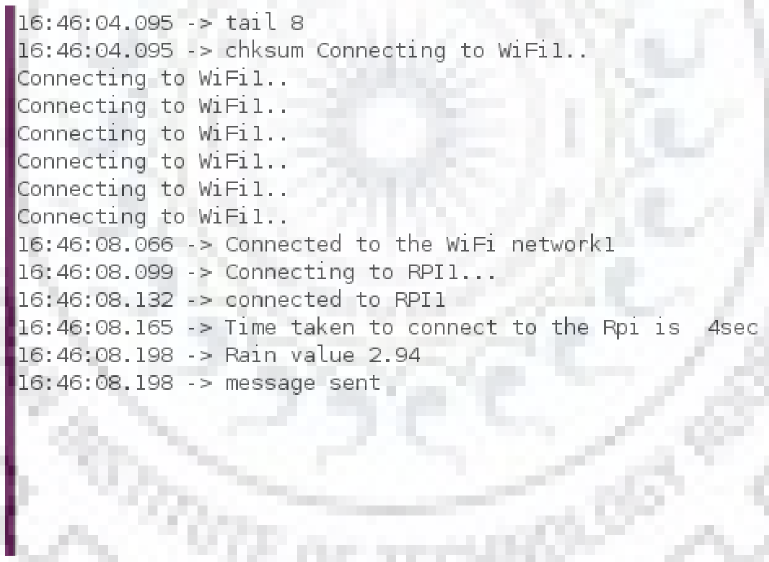
When the link between the edge server and main server is not recovered for a long time, cloud server is unable to receive the rainfall data. This prevents it to store the data of that time interval. Hence, every time there is a disconnection, there is a penalty of data loss which affects the training module of machine learning model. It is an incremental machine learning model, so training is done after every fixed interval and model file is updated. As the ML model uses antecedent rainfall analysis, so when the connection will be regained, there would be no record of the period when there was no connection. The model is dependent on antecedent rainfall value of last 30 days. So when new data comes after reconnection, missing data somewhat effects the accuracy of the model and hence prediction.

4.3.6 Overhead of transferring Model file to the Edge server

As we explained in our proposed system, server trains the machine learning model with the rainfall data and tests the incoming values using the trained model. This training activity is run after every fixed interval and then the model file is sent to the edge servers. Each edge server receives the trained model file specific to its zone. With the use of that trained model file, edge servers test the rain data and predicts landslide. Hence, in order to make the edge smart and do some processing, it takes a little overhead of sending the model file to the edge servers.

4.3.7 SCREENSHOTS

The snapshots of various scenarios have been presented in this section :

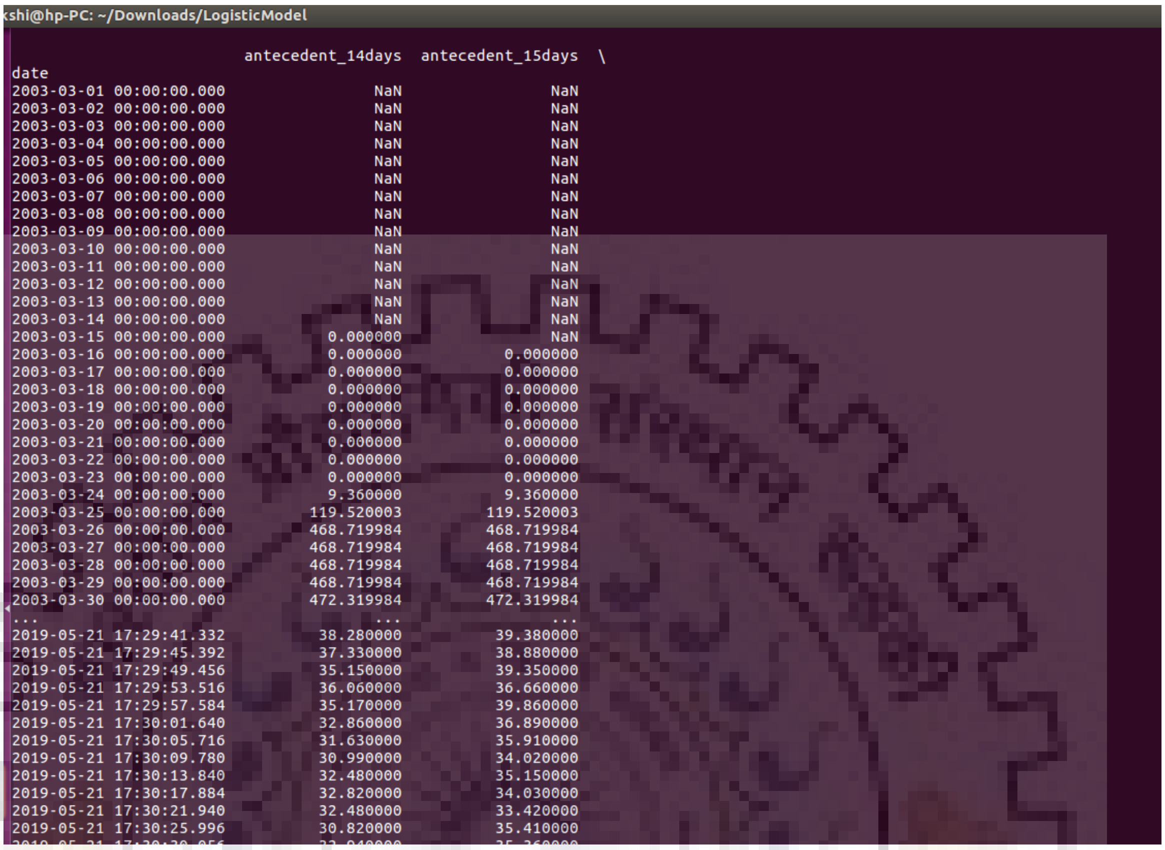


```
16:46:04.095 -> tail 8
16:46:04.095 -> chksum Connecting to WiFi1..
Connecting to WiFi1..
Connecting to WiFi1..
Connecting to WiFi1..
Connecting to WiFi1..
Connecting to WiFi1..
Connecting to WiFi1..
16:46:08.066 -> Connected to the WiFi network1
16:46:08.099 -> Connecting to RPI1...
16:46:08.132 -> connected to RPI1
16:46:08.165 -> Time taken to connect to the Rpi is  4sec
16:46:08.198 -> Rain value 2.94
16:46:08.198 -> message sent
```

FIGURE 4.7: Connecting NodeMCU to Raspberry Pi

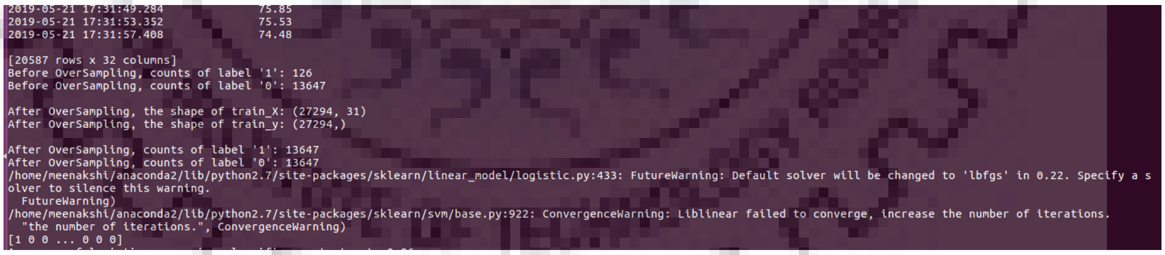
4.4 Summary

This Chapter talks about the laboratory setup to demonstrate the working proposed model. It lists the hardware devices along with the communication protocols that have been



```
kshi@hp-PC: ~/Downloads/LogisticModel

                         antecedent_14days  antecedent_15days  \
date
2003-03-01 00:00:00.000                NaN                NaN
2003-03-02 00:00:00.000                NaN                NaN
2003-03-03 00:00:00.000                NaN                NaN
2003-03-04 00:00:00.000                NaN                NaN
2003-03-05 00:00:00.000                NaN                NaN
2003-03-06 00:00:00.000                NaN                NaN
2003-03-07 00:00:00.000                NaN                NaN
2003-03-08 00:00:00.000                NaN                NaN
2003-03-09 00:00:00.000                NaN                NaN
2003-03-10 00:00:00.000                NaN                NaN
2003-03-11 00:00:00.000                NaN                NaN
2003-03-12 00:00:00.000                NaN                NaN
2003-03-13 00:00:00.000                NaN                NaN
2003-03-14 00:00:00.000                NaN                NaN
2003-03-15 00:00:00.000           0.000000                NaN
2003-03-16 00:00:00.000           0.000000           0.000000
2003-03-17 00:00:00.000           0.000000           0.000000
2003-03-18 00:00:00.000           0.000000           0.000000
2003-03-19 00:00:00.000           0.000000           0.000000
2003-03-20 00:00:00.000           0.000000           0.000000
2003-03-21 00:00:00.000           0.000000           0.000000
2003-03-22 00:00:00.000           0.000000           0.000000
2003-03-23 00:00:00.000           0.000000           0.000000
2003-03-24 00:00:00.000           9.360000           9.360000
2003-03-25 00:00:00.000         119.520003         119.520003
2003-03-26 00:00:00.000         468.719984         468.719984
2003-03-27 00:00:00.000         468.719984         468.719984
2003-03-28 00:00:00.000         468.719984         468.719984
2003-03-29 00:00:00.000         468.719984         468.719984
2003-03-30 00:00:00.000         472.319984         472.319984
...                                    ...                ...
2019-05-21 17:29:41.332          38.280000          39.380000
2019-05-21 17:29:45.392          37.330000          38.880000
2019-05-21 17:29:49.456          35.150000          39.350000
2019-05-21 17:29:53.516          36.060000          36.660000
2019-05-21 17:29:57.584          35.170000          39.860000
2019-05-21 17:30:01.640          32.860000          36.890000
2019-05-21 17:30:05.716          31.630000          35.910000
2019-05-21 17:30:09.780          30.990000          34.020000
2019-05-21 17:30:13.840          32.480000          35.150000
2019-05-21 17:30:17.884          32.820000          34.030000
2019-05-21 17:30:21.940          32.480000          33.420000
2019-05-21 17:30:25.996          30.820000          35.410000
```

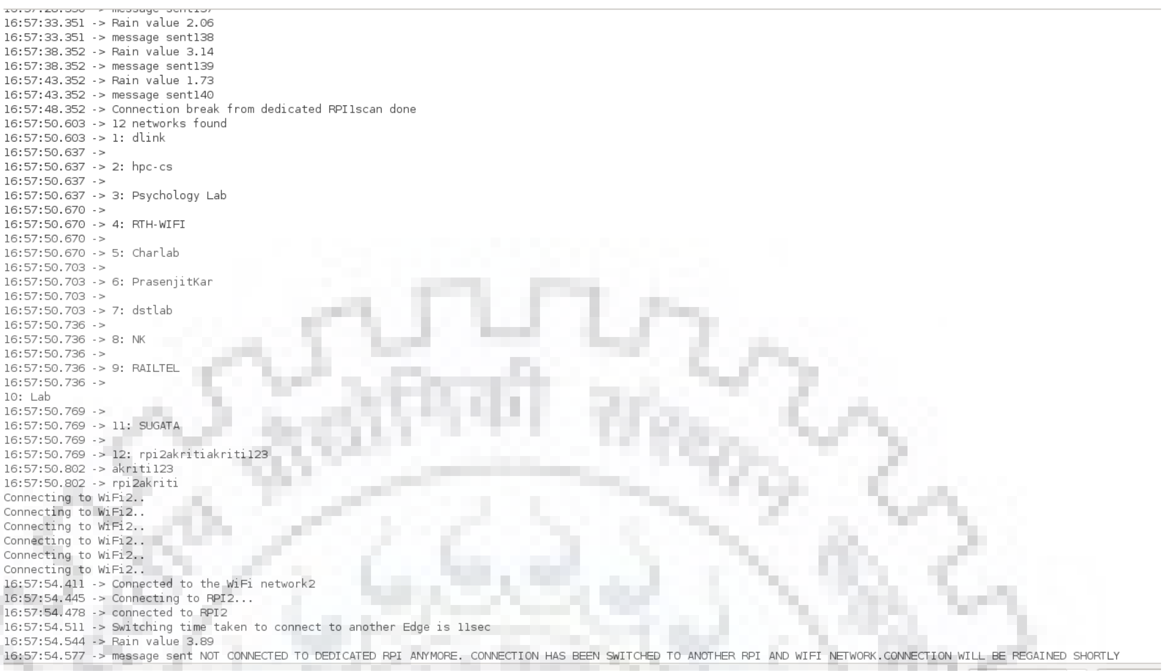
FIGURE 4.8: Antecedent Rainfall Calculation



```
2019-05-21 17:31:49.284              75.85
2019-05-21 17:31:53.352              75.53
2019-05-21 17:31:57.408              74.48

[20587 rows x 32 columns]
Before OverSampling, counts of label '1': 126
Before OverSampling, counts of label '0': 13647

After OverSampling, the shape of train_X: (27294, 31)
After OverSampling, the shape of train_y: (27294,)

After OverSampling, counts of label '1': 13647
After OverSampling, counts of label '0': 13647
/home/meenakshi/anaconda2/lib/python2.7/site-packages/sklearn/linear_model/logistic.py:433: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a s
olver to silence this warning.
  FutureWarning)
/home/meenakshi/anaconda2/lib/python2.7/site-packages/sklearn/svm/base.py:922: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
[1 0 0 ... 0 0 0]
```

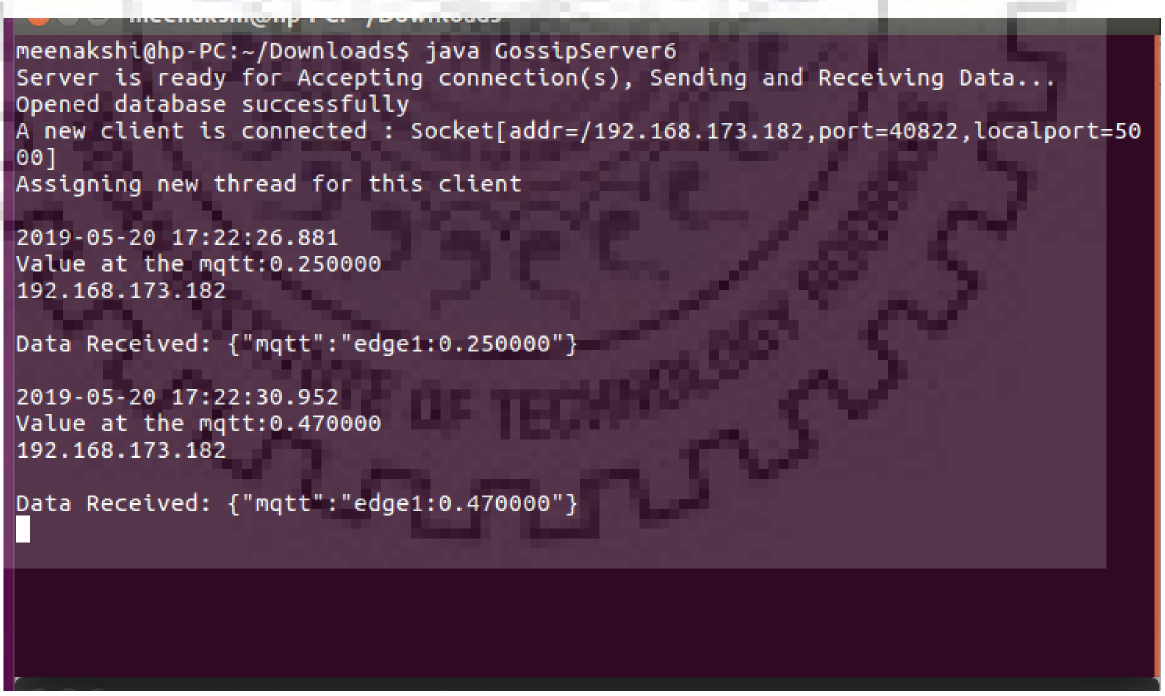
FIGURE 4.9: Prediction on data values

used for the experimental purpose in the lab. This section also explains the machine learning algorithm that has been developed for predicting landslides. Further, various parameters have been talked about : Recovery time, impact on accuracy of ML algorithm due to disconnections. Also main server and Edge server are compared on the basis of Execution



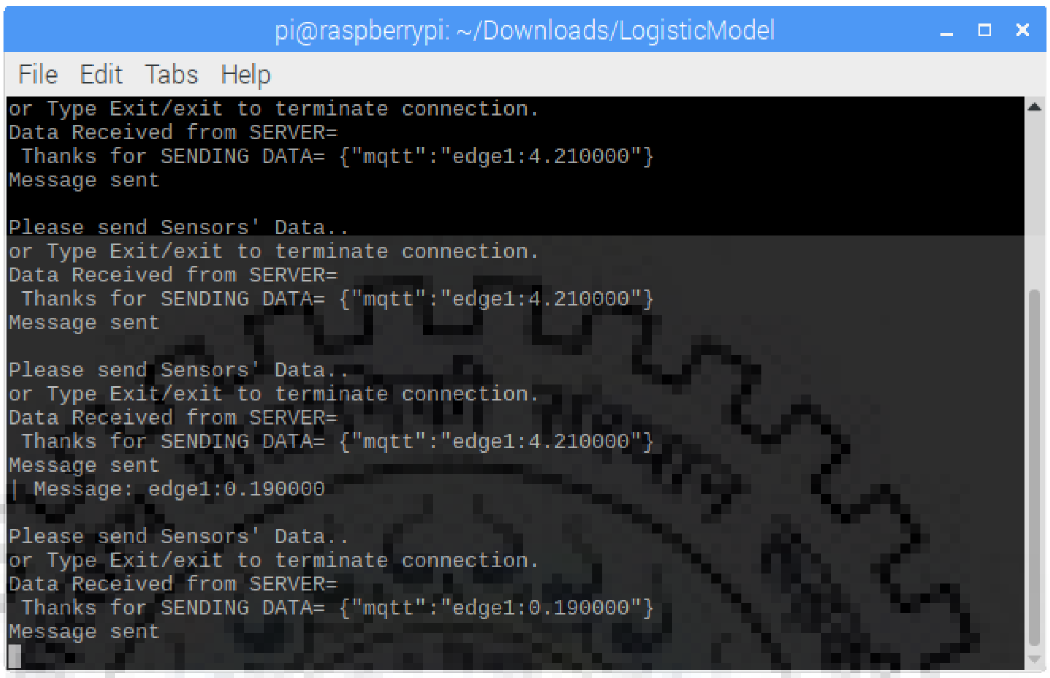
```
16:57:33.351 -> Rain value 2.06
16:57:33.351 -> message sent138
16:57:38.352 -> Rain value 3.14
16:57:38.352 -> message sent139
16:57:43.352 -> Rain value 1.73
16:57:43.352 -> message sent140
16:57:48.352 -> Connection break from dedicated RPI1scan done
16:57:50.603 -> 12 networks found
16:57:50.603 -> 1: dlink
16:57:50.637 ->
16:57:50.637 -> 2: hpc-cs
16:57:50.637 ->
16:57:50.637 -> 3: Psychology Lab
16:57:50.670 ->
16:57:50.670 -> 4: RTH-WIFI
16:57:50.670 ->
16:57:50.670 -> 5: Charlab
16:57:50.703 ->
16:57:50.703 -> 6: PrasenjitKar
16:57:50.703 ->
16:57:50.703 -> 7: dstlab
16:57:50.736 ->
16:57:50.736 -> 8: NK
16:57:50.736 ->
16:57:50.736 -> 9: RAILTEL
16:57:50.736 ->
10: Lab
16:57:50.769 ->
16:57:50.769 -> 11: SUGATA
16:57:50.769 ->
16:57:50.769 -> 12: rpi2akritiakriti123
16:57:50.802 -> akriti123
16:57:50.802 -> rpi2akriti
Connecting to WiFi2..
Connecting to WiFi2..
Connecting to WiFi2..
Connecting to WiFi2..
Connecting to WiFi2..
Connecting to WiFi2..
16:57:54.411 -> Connected to the WiFi network2
16:57:54.445 -> Connecting to RPI2...
16:57:54.478 -> connected to RPI2
16:57:54.511 -> Switching time taken to connect to another Edge is 11sec
16:57:54.544 -> Rain value 3.89
16:57:54.577 -> message sent NOT CONNECTED TO DEDICATED RPI ANYMORE. CONNECTION HAS BEEN SWITCHED TO ANOTHER RPI AND WIFI NETWORK.CONNECTION WILL BE REGAINED SHORTLY
```

FIGURE 4.10: Connection switch to other Raspi in case of loss to edge server



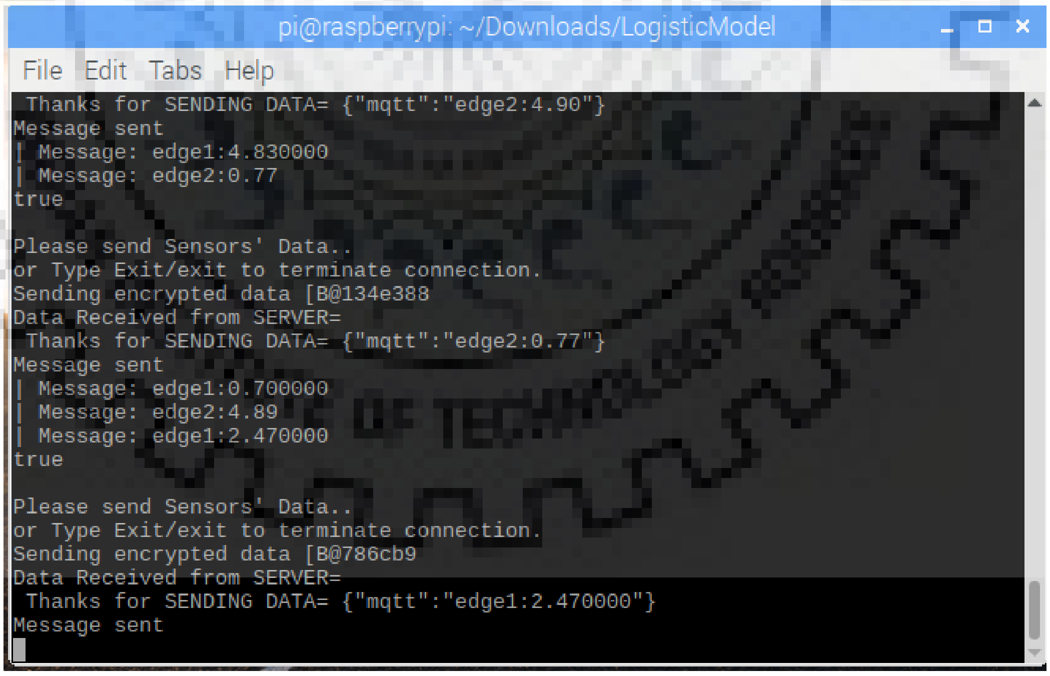
```
meenakshi@hp-PC:~/Downloads$ java GossipServer6
Server is ready for Accepting connection(s), Sending and Receiving Data...
Opened database successfully
A new client is connected : Socket[addr=/192.168.173.182,port=40822,localport=50
00]
Assigning new thread for this client

2019-05-20 17:22:26.881
Value at the mqtt:0.250000
192.168.173.182

Data Received: {"mqtt":"edge1:0.250000"}

2019-05-20 17:22:30.952
Value at the mqtt:0.470000
192.168.173.182

Data Received: {"mqtt":"edge1:0.470000"}
```

FIGURE 4.11: Server collecting rainfall data from one of the edge



```
or Type Exit/exit to terminate connection.
Data Received from SERVER=
 Thanks for SENDING DATA= {"mqtt":"edge1:4.210000"}
Message sent

Please send Sensors' Data..
or Type Exit/exit to terminate connection.
Data Received from SERVER=
 Thanks for SENDING DATA= {"mqtt":"edge1:4.210000"}
Message sent

Please send Sensors' Data..
or Type Exit/exit to terminate connection.
Data Received from SERVER=
 Thanks for SENDING DATA= {"mqtt":"edge1:4.210000"}
Message sent
| Message: edge1:0.190000

Please send Sensors' Data..
or Type Exit/exit to terminate connection.
Data Received from SERVER=
 Thanks for SENDING DATA= {"mqtt":"edge1:0.190000"}
Message sent
```

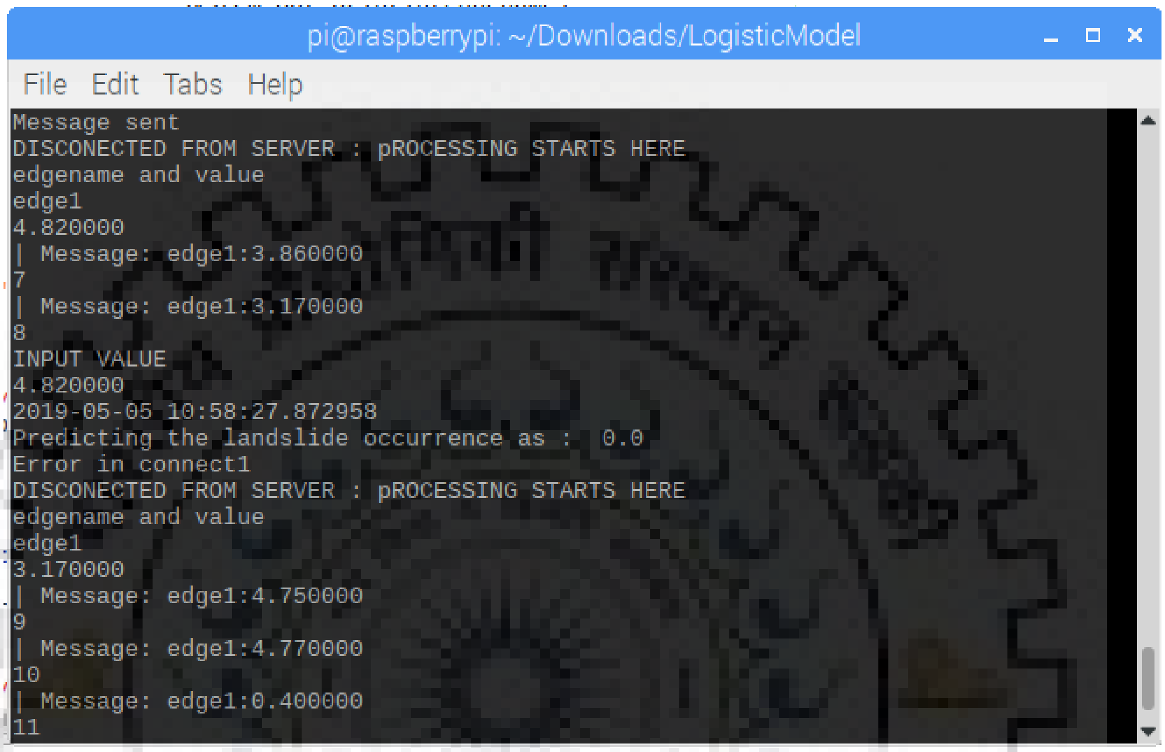
FIGURE 4.12: Raspberry Pi subscribing to the Nodemcu and forwarding data to server



```
 Thanks for SENDING DATA= {"mqtt":"edge2:4.90"}
Message sent
| Message: edge1:4.830000
| Message: edge2:0.77
true

Please send Sensors' Data..
or Type Exit/exit to terminate connection.
Sending encrypted data [B@134e388
Data Received from SERVER=
 Thanks for SENDING DATA= {"mqtt":"edge2:0.77"}
Message sent
| Message: edge1:0.700000
| Message: edge2:4.89
| Message: edge1:2.470000
true

Please send Sensors' Data..
or Type Exit/exit to terminate connection.
Sending encrypted data [B@786cb9
Data Received from SERVER=
 Thanks for SENDING DATA= {"mqtt":"edge1:2.470000"}
Message sent
```

FIGURE 4.13: Edge 1 receiving data from Edge 2 after connection switching and hence giving uninterrupted transmission

time and CPU workload. Statistical analysis of CPU workload and Packet drop percentage has been made. Further, screenshots are attached to support the work done.



```
Message sent
DISCONECTED FROM SERVER : pROCESSING STARTS HERE
edgename and value
edge1
4.820000
| Message: edge1:3.860000
7
| Message: edge1:3.170000
8
INPUT VALUE
4.820000
2019-05-05 10:58:27.872958
Predicting the landslide occurrence as :  0.0
Error in connect1
DISCONECTED FROM SERVER : pROCESSING STARTS HERE
edgename and value
edge1
3.170000
| Message: edge1:4.750000
9
| Message: edge1:4.770000
10
| Message: edge1:0.400000
11
```

FIGURE 4.14: Prediction Algorithm being run on the edge in case of connection loss to server

5 Conclusions

5.1 Conclusion

The proposed system is expected to make the IoT – cloud architecture as reliable as possible. Even if the connection to main server would be down for a while , the edge server would be readily available for processing the current incoming data from the IoT node and raise warning alarms in case of discrepancy. The system is also expected to handle the situation where the particular Edge server is not accessible by switching the connection to another available edge in the range. In this manner, we can give uninterrupted service at the edge level. So, data analysis can happen both at the Cloud server and Edge server according to the circumstances and hence increase the system's reliability. The proposed system can be implemented with any other real-time IoT based systems which demand high service reliability and delay intolerance.

5.2 Future Work

This work is a prototype developed under NMHS project to make the LEWS system as reliable as possible by introducing the concept of Edge. For experimental purpose, the equipment used in the project are cost friendly. As an alternative to nodeMCU and raspberry pi, more powerful devices can be used adhering to the overall project budget which can further optimize the network range, accuracy and efficiency of the designed system. Hence, the developed model can be expanded further to implement it for real time scenario.

Reference

[1] Gubbi, Jayavardhana, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. ”Internet of Things (IoT): A vision, architectural elements, and future directions.” Future generation computer systems 29, no. 7 (2013): 1645-1660.

[2] Breivold, Hongyu Pei, and Kristian Sandström. ”Internet of things for industrial automation–challenges and technical solutions.” In 2015 IEEE International Conference on Data Science and Data Intensive Systems. IEEE, 2015, pp. 532-539.

[3] Hegazy, Tamir, and Mohamed Hefeeda. ”Industrial automation as a cloud service.” IEEE Transactions on Parallel and Distributed Systems 26.10 (2014): 2750-2763.

[4] Chaturvedi, Pratik, Brajesh Jaiswal, Sumit Sharma, and Neetu Tyagi. ”Instrumentation Based Dynamics Study of Tangni Landslide near Chamoli, Uttrakhand.” International Journal of Research in Advent Technology 2, no. 10 (2014).

[5] Satyanarayanan, Mahadev. ”The emergence of edge computing.” Computer 50, no. 1 (2017): 30-39.

[6] Paul, Anand, Hameed Pinjari, Won-Hwa Hong, Hyun Cheol Seo, and Seungmin Rho. ”Fog computing-based IoT for health monitoring system.” Journal of Sensors 2018 (2018).

[7] Yu Cao, Songqing Chen, Peng Hou and D. Brown, ”FAST: A fog computing assisted distributed analytics system to monitor fall for stroke mitigation,” 2015 IEEE International Conference on Networking, Architecture and Storage (NAS), Boston, MA, 2015, pp. 2-11.

[8] Q. Liu and T. Han, ”DARE: Dynamic Adaptive Mobile Augmented Reality with Edge Computing,” 2018 IEEE 26th International Conference on Network Protocols (ICNP), Cambridge, United Kingdom, 2018, pp. 1-11.

[9] Zamora-Izquierdo, Miguel A., José Santa, Juan A. Martínez, Vicente Martínez, and Antonio F. Skarmeta. ”Smart farming IoT platform based on edge and cloud computing.” Biosystems Engineering 177 (2019): 4-17.

[10] Ashouri, Majid, Paul Davidsson, and Romina Spalazzese. ”Cloud, Edge, or Both? Towards Decision Support for Designing IoT Applications.” The Fifth International Conference on Internet of Things: Systems, Management and Security (IoTSMS 2018).2018, pp. 155-162.

[11] Le, Minh, Zheng Song, Young-Woo Kwon, and Eli Tilevich. ”Reliable and efficient mobile edge computing in highly dynamic and volatile environments.” In 2017 Second International Conference on Fog and Mobile Edge Computing (FMEC). IEEE, 2017, pp. 155-162.

[12] Liu, Jianhui, and Qi Zhang. ”Offloading schemes in mobile edge computing for ultra-reliable low latency communications.” Ieee Access 6 (2018): 12825-12837.

[13] Tareq, Md Mostofa Kamal, Omid Semiari, Mohsen Amini Salehi, and Walid Saad. ”Ultra Reliable, Low Latency Vehicle-to-Infrastructure Wireless Communications with Edge Computing.” In 2018 IEEE Global Communications Conference (GLOBECOM). IEEE, 2018, pp. 1-7.

[14] Javed, Asad, Keijo Heljanko, Andrea Buda, and Kary Främling. ”Cefiot: A fault-tolerant iot architecture for edge and cloud.” In 2018 IEEE 4th World Forum on Internet of Things (WF-IoT). IEEE, 2018, pp. 813-818.

[15] Bai, Shibiao, Jian Wang, Benni Thiebes, Chen Cheng, and Yipeng Yang. ”Analysis of the relationship of landslide occurrence with rainfall: a case study of Wudu County, China.” Arabian Journal of Geosciences 7, no. 4 (2014): 1277-1285.

[16] Kodali, Ravi Kishore, and Kopulwar Shishir Mahesh. ”A low cost implementation of MQTT using ESP8266.” In 2016 2nd International Conference on Contemporary Computing and Informatics (IC3I). IEEE, 2016, pp. 404-408.

[17] Jain, Sarthak, Anant Vaibhav, and Lovely Goyal. ”Raspberry Pi based interactive home automation system through E-mail.” In 2014 International Conference on Reliability Optimization and Information Technology (ICROIT). IEEE, 2014, pp. 277-280.