FAULT TOLERANCE IN MOBILE CROWDSOURCING

Project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology

In

Computer Science & Engineering and Information Technology

By

Rohan Kanoo (141408)

Mudit Bansal (141418)

Under the supervision of

Dr. Geetanjali Rathee

To

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

Candidate's Declaration

I hereby declare that the work presented in this report entitled **"Fault Tolerance in Mobile Crowdsourcing"** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from July 2017 to May 2018 under the supervision of **Dr. Geetanjali Rathee** (Associate Professor, Computer science and Engineering).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Student Signature)	(Student Signature)
Rohan Kanoo, 141408	Mudit Bansal, 141418

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

(Supervisor Signature)
Dr. Geetanjali Rathee
Associate Professor
CSI/IT
Dated:

ACKNOWLEDGEMENT

I have taken efforts into this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

I am highly indebted to **Dr. Geetanjali Rathee** for her guidance and constant supervision as well as for providing necessary information regarding the project & for her support in completing the project.

I would like to express my gratitude towards my member of Jaypee University of Information Technology for their kind co-operation and encouragement, which help me in completion of this project.

My thanks and appreciations go to my colleague in developing the project and people who have willingly helped me out with their abilities.

TABLE OF CONTENTS

LIST OF ABBRIVIATIONS

1. **MCN –** Mobile Crowdsourcing Network
2. **ITA –** Improved Two-stage Auction algorithm
3. **TORU –** Truthful Online Reputation Updating algorithm
4. **GPS –** Global Positioning System
5. **TBA –** Threshold Based Auction
6. **TOIM –** Truthful Online Incentive Mechanism
7. **MRT –** Mass Rapid Transit
8. **SMS –** Short Message Service
9. **API –** Application Program Interface

LIST OF FIGURES

Page No.

Abstract

While defining Mobile Crowdsourcing Network (MCN) we can say that it is a very promising computer architecture. To perform tasks in this type of system the involvement of human and powerful mobile devices having high computation power is the key principle that is applied in the crowdsourcing. However, there are many issues with the above mentioned system like some critical security and privacy issues that hinders the MCN. To understand the issues we will first understand the general architecture of the MCN, which generally comprising of two things, namely crowdsourcing sensing and crowdsourcing computing. After understanding the basics of this system, we will be implementing an auction algorithm and simultaneously tolerating the fault.

Keeping in mind the end goal we enhance the effectiveness and utility of mobile crowdsourcing systems, we will propose a motivating incentive mechanism with privacy protection and fault tolerance mechanism in mobile crowdsourcing system. When we combine the advantages of online and offline incentive mechanism, the resultant mechanism that will be selecting statically the worker nominee, and will be selecting the winners dynamically, after the workers have submitted the bidding, while tolerating any fault that could arise due to database connectivity. The system that we proposed consists of two algorithms, namely Improved Two-stage Auction algorithm (ITA) and Truthful Online Reputation Updating algorithm (TORU)

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION

Crowdsourcing is a specific sourcing model in which organizations use contributions from internet users to obtain needed services or ideas. It can be of great use in areas such as entrepreneurship, science and health, education, etc. The term crowdsourcing was devised in the year 2005 as a combination of two words i.e. crowd and outsourcing. This way of sourcing, where we divide the work between participants to achieve a combined result, was already a successful phenomenon before the digital age. Crowdsourcing is different from outsourcing because in crowdsourcing the work come from public that is mostly undefined (rather than being from a commissioned and specific, named group) and the crowdsourcing is a mixture of bottom-up process and top-down processes. Using crowdsourcing has its own advantages that include quality, scalability, speed, diversity, flexibility and improved costs. Crowdsourcing provides a way for organization so that they could learn beyond what their “base of minds” of employees provides in the form of innovation contests or idea competitions. Other tedious “micro task” can also be involved in crowdsourcing, which is performed by large crowd in parallel, also known as paid crowd for example Amazon Mechanical Turk. We can also use crowdsourcing for non-commercial work and to develop common goods for example Wikipedia. In crowdsourcing contexts when we have to evaluate the performance of ideas, the user communication and the platform presentation effects should be taken into account. To better understand crowdsourcing, Fig 1.1 depicts the working of a crowdsourcing system.

Any activity than can take place on smartphones or mobile platforms involves mobile crowdsourcing if it uses GPS technology frequently. This results for data gathering in real-time and gives greater accessibility and reach to projects. Crowdsourcing activities are processed on mobile phones or other handheld mobile devices. However, mobile crowdsourcing can result to an urban bias, as well as privacy and safety concerns.

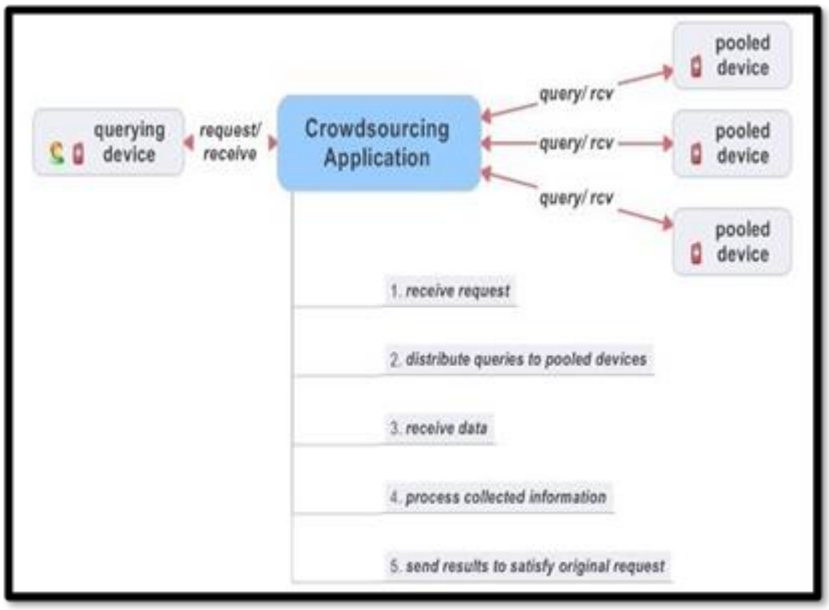

querying device
request/ receive
Crowdsourcing Application
query/ rcv
query/ rcv
query/ rcv
pooled device
pooled device
pooled device
1. receive request
2. distribute queries to pooled devices
3. receive data
4. process collected information
5. send results to satisfy original request

Fig1.1 – Working of Crowdsourcing Application

Fault tolerance is the property that enables a system to continue operating properly, possibly at a reduced level, even if there is a failure of (or one or more faults within) it's some of the components. If there is a huge decrease in the operating quality, the decrease is proportional to the severity of the failure occurred, as compared to a system which is newly designed in which even if a small failure occur it can cause a total breakdown. Fault tolerance is especially used in high-availability or life-critical systems. In addition, it is used in virtual machines that can be seen in Fig 1.2. The ability of maintaining functionality when portions of a system break down is referred to as graceful degradation. The term is most commonly used to describe computer systems designed to continue, more or less, fully operational with, perhaps, a reduction in throughput or an increase in response time in the event of some partial failure. A basic computer design with fault tolerance ability is shown in Fig 1.3.

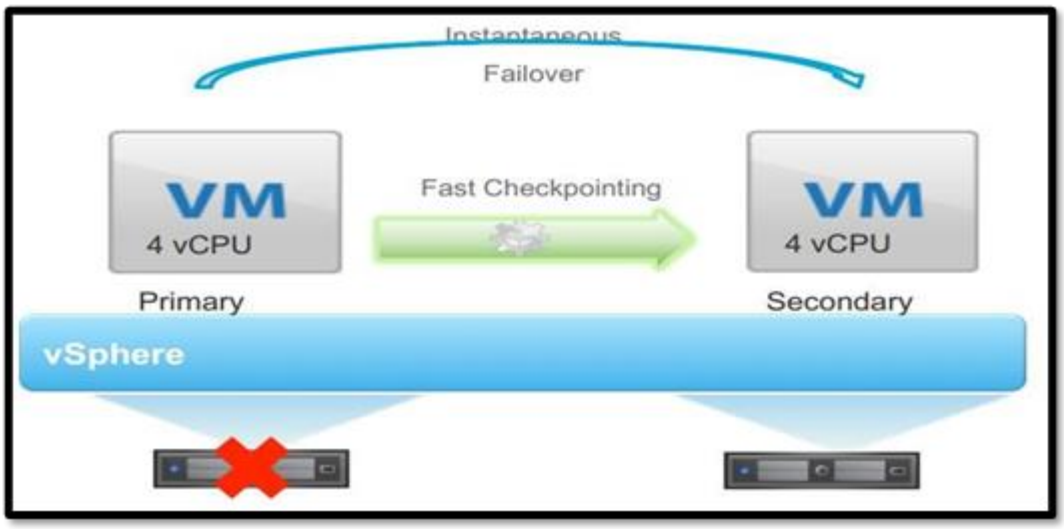


Fig 1.2 – Fault Tolerance in Virtual Machines

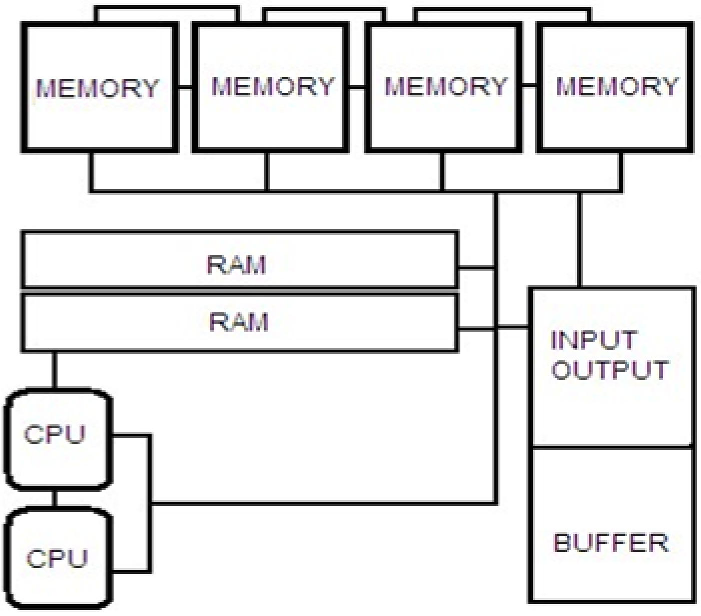


Fig 1.3 – Fault-tolerant computer design

1.2 MOBILE CROWDSOURCING ARCHITECTURE

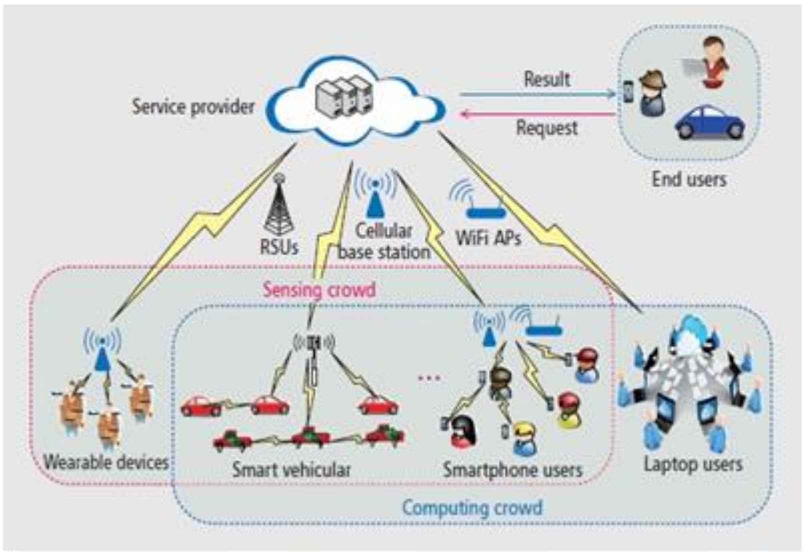


Fig 1.4 – General Architecture of Mobile Crowdsourcing Network

Architecture of mobile crowdsourcing is depicted in Fig 1.4 and explained below with components belonging to each entity of mobile crowdsourcing network.

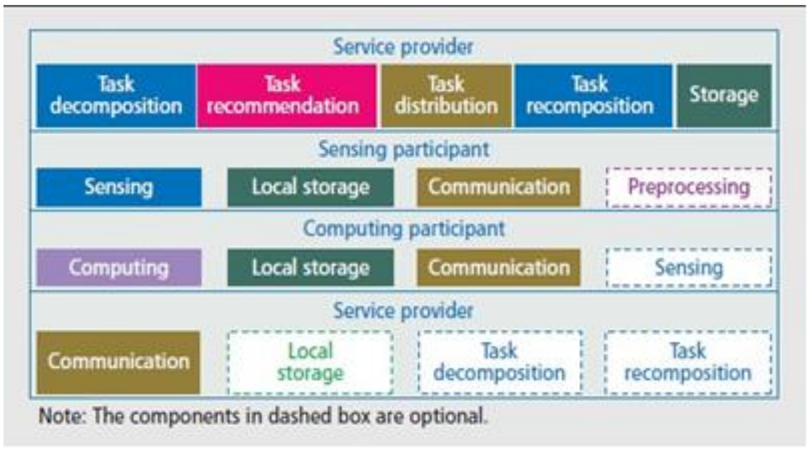


Fig 1.5 – Components belonging to each entity in Mobile Crowdsourcing Network

1.2.1 Server or Service Provider

Servers or Service Provider is a crowdsourcing platform, which provides services to both final stage users and sensing & computing crowd. Service provider accepts task requests from final stage users and publish them on its service board for mobile users. As shown in Fig 1.5, service provider has following components:

- *Task Decomposition and Task Recomposition* component which helps in partitioning the tasks before publishing them to crowd and combining all the results before sending it to service consumers respectively. After receiving the results from the crowd, service provider does some processing and give back results to end users.

- *Task Distribution* component helps in distributing tasks to crowd after decomposing them. End users may choose to decompose the task themselves, therefore Service Provider has the job to distribute those tasks.

- *Task Recommendation* component that gives opportunity to crowd to submit their subscription trapdoors to give their preferences on crowdsourced tasks. Whenever there is a crowdsourced task, this component checks if crowd preferences match with task, if it matches it send alert to that mobile user.

Sensing and Computing are the main two types of crowdsourcing tasks. Sensing tasks' main work is to gather sensed and important data from the sensors of mobile phones of users or crowd who carry such type of devices. These tasks return the sensed data from sensing crowd to servers, which can store that data or can send it to end users. Computing tasks are mainly designed to crowdsource the computation part (usually require human intelligence) to large number of participants having mobile computing devices e.g. tablets, mobile phones etc.

Service Provider is honest but may refer some sensed data having personal information (identity, location information, interests etc.), published tasks, computed data or results, which can led to some privacy related issues discussed later.

1.2.2 End Users and Service Consumers

These are the customers who require the cloud services through mobile crowdsourcing system. They can purchase or rent crowdsourcing services at some cost and utilize cloud services by outsourcing tasks to mobile users. Service requests are sent to service providers and receive results from that only. Therefore, end users should have communication components as shown in Fig 1.5, in order to put tasks and receive results. They themselves can decompose and recompose the task to be published and dependent on service provider only for publishing those tasks to crowd. End users can also participate in completing other crowdsourced tasks and behave like sensing or computing crowd. Security assumption for both end users and crowd should be same.

1.2.3 Crowd Sensing Participants

These are the participants who participate in crowdsourced sensing tasks i.e. perform data sensing with sensor equipped mobile devices. To complete these tasks, sensing crowd should have some sensing components, communication components and local storage components. Sensing components can be microphones, camera, GPS, accelerometers etc. Some of the sensing devices can have pre-processing component so as to pre-process the sensed data. Sensing crowd cannot be trusted always e.g. some of the sensing participants can be malicious and report some invalid data to service providers and can also launch Denial of Service Attack which is, they accept all the tasks without giving back any results.

1.2.4 Crowd Computing Participants

These are the participants who participate in crowdsourced computing tasks. Computing tasks are of two types:

1. **Sensing based computing task:** These tasks take input as data from end users or service providers and also some current sensed data (from their sensor enabled devices). Here, sensed data is directly used as input to computation part rather than sending it to service providers.
2. **Pure Computing Task:** These tasks take input (task or data) from service provider and perform some computations and return back the results. Therefore, these pure computing crowd need to have only computing devices i.e. mobile phones, tablets etc.

Computing devices should have computing components, local storage component and communication components. Computing crowd is not always trusted as some malicious users may send back some wrong results.

1.3 MOTIVATION IN CROWDSOURCING

Crowdsourced tasks involve human and it is necessary to motivate crowd to complete crowdsourced tasks. Crowd can be intrinsically or extrinsically motivated. Below shows categorization of motivation w.r.t. crowdsourced tasks:

1.3.1 Intrinsic Motivation

In this case, person is active and its attempt to complete that task is not derived by any monetary rewards, he may be active just for fun or task completion makes him feel happy. It is categorized into Enjoyment based or Community based as described below:

A. Enjoyment Based

This leads to sensation of fun to workers. Few examples are:

- *Skill Variety:* A worker may pick translation task because he likes translating and wants to use his skills in some particular language. Therefore, he will be motivated more if higher variety of skills are available which are of his/her interest.

- *Task Autonomy:* A worker may be interested in tasks which gives him some degree of freedom (own decisions and creativity permitted) during task execution e.g. designing a logo or website.

- *Direct feedback from the job:* User is motivated because it gives him opportunity to check results of performed tasks e.g. a programming task.

- *Pastime:* Workers are motivated, as they want to avoid their boredom.

B. Community Based

Community based motivation comes into picture where participation of workers is guided by community. Few examples are:

- *Community Identification:* A worker is motivated to complete tasks as task requesters has good reputation.
- *Social Contact*: A person is participating, as he wants to meet new people through crowdsourcing platform.

1.3.2 Extrinsic Motivation

In this case, person is active as his attempt to complete task helps in achieving some desired outcome e.g. money. The effectiveness of mobile crowdsourcing systems depends on the worker's mentality that affects user participation and completing those published tasks. Participation levels can be improved by providing incentives (rewards such as monetary, social, gaming related etc.). Increased user participation can help in extending the system coverage and also enhances the quality of collected data. Therefore, there should be some approach to include incentives in projects. It is categorized into:

A. Immediate Payoffs

Workers are motivated as completion of these tasks helps receiving immediate compensation (Payment) for that work. Monetary remuneration can act as primary or secondary source of income for that worker.

B. Delayed Payoffs

These type of tasks can be used to generate future material advantages e.g. worker is motivated in completing translation tasks as he wants to improve his language skills for new and better job or a worker may be selecting some tasks in order to show his presence and noticed by employers.

C. Social Motivation

It is extrinsic counterpart of intrinsic community based motivation.

- *Action significance by external obligations and norms:* Person is motivated by some third party from outside the platform, e.g. a student is working on some task as he is told by his teacher to do so.
- *Indirect Feedback from job:* A worker is motivated as it helps him in getting formal praise from other people.

Incentive Types refers to type of reward e.g. monetary, social or gaming related to be offered to user. *Incentive Magnitude* refers to amount of each reward offered. *Incentive Structure* refers to the structure which helps in governing the distribution of various incentives to users e.g. performance metrics etc. Two incentive mechanisms are:

Reward sharing, in which fixed rewards are offered by service consumers for outsourced tasks and shared among the task participants according to the time they have worked. **Auction Based**, in which mobile users make offers for different crowdsourced tasks and service consumers choose mobile users in order to maximize their own utilities.

1.4 PROBLEM STATEMENT

We want to secure the personal data of people participating in crowdsourcing and tolerate the fault that could have occurred due to failure of database. Because, there may be a situation when database is connectivity is lost or there is a problem connecting to the database also, there is sensitive data involved of the user, it is important to protect the users from malicious and unauthorized attack of their personal data. Every day we hear news of data leak and privacy being compromised of many people. If this problem is not addressed properly, in crowdsourcing, people will start losing trust in the process. This will lead to less people participating in crowdsourcing thus the end user or the requester will have to pay more to a single or small group of people. In addition, the company requesting ideas from the crowd, for their product, would not get a good amount of feedback and the product will be not up to the mark. To prevent malicious attack, unauthorized access and manipulation of crowdsourced data and private data of crowd and to tolerate fault, we will use Improved Two- stage Auction algorithm (ITA) with multiple threading using akka API.

1.5 OBJECTIVES

Reliability – To focus on a continuous service of a distributed system without any interruptions. Designing system such that the fault tolerant ability of the system is high. The system should be reliable enough so that it could process the raw data collected from the crowd in same manner and with same precision.

Availability – To keep system ready so that it can be access anytime. Need of system can arise anytime, so it should be up and running whenever the end user request for it. In addition, the same system is being used by the crowd to send their data so if somehow system is shutdown it will result in loss of data.

Security – Security is the main concern of mobile crowdsourcing. With so many sensors and sensing devices present in a typical mobile phone, the device constantly pick up sensitive and private data of users like location, passwords, etc. So, there is a constant risk of these data being leaked or compromised thus, to prevent any unauthorized access of the system we will use algorithms and incentives mechanism.

1.6 METHODOLGY

In this project, the main focus will be on finalizing the workers who will be able to complete the outsourced task at minimum cost and at the earliest. This will be achieved by using Improved Two-way Auction (ITA) algorithm that helps in shortlisting the useful workers for the task. Also, we will be using the akka API of java to divide the algorithm to different thread so that we could tolerate fault if there arise any problem like error in connecting to the database. In addition, we will try to solve one more problem in mobile crowdsourcing known as the free-riding problem, which is of major concern.

CHAPTER 2
LITERATURE SURVEY

In recent years, smartphone market has expanded considerably, and is expected to keep on expanding in the near future. By mobile crowdsourcing, we mean to practice crowdsourcing on mobile devices or smartphones. It’s easy for mobile phone users to work on crowdsourcing assignments easily because of the innovative and improved software and hardware in the mobile phones for example high precision GPS, high resolution camera, etc. Nowadays, mobile crowdsourcing is more than simple portrayals of site. It is easy to locate and create precise movement profiles with help of users who own a smartphones equipped with GPS. When smartphones users upload information like precise location or data about menu of a restaurant it is known as dynamic crowdsourcing. Also, in an event of disaster we can use mobile crowdsourcing to record damage situations and safeguard people by sending coordinates to rescuers. Some advantage of mobile crowdsourcing is data collected is up-to-date and exact and delivery of information is rapid.

Mobile crowdsourcing system can be thought as of commercial crowdsourcing system. Amazon Mechanical Turk which is a customary commercial crowdsourcing systems. This system characterize how much the workers will be paid per assignment for the submitted task and workers need to give in a confirmation of the finished task. Some random workers within the crowd submit the required verification, after working on the task, to the crowdsourcing platform. When the worker send the verification report to the employee, the employee pay the worker if the task is completed successfully. But, it is not compulsory to always accept the worker. The workers keep on coming and bidding for a task and it’s up to the system either to accept or deny the worker in least time possible. So, in order to fetch better advantage of the system the worker in mobile crowdsourcing are chosen using a bidding procedure. Mobile phones sensing have great capabilities. There has been numerous research to develop different applications and systems for example VTrack for giving real time traffic information, Sensorly to make cellular network coverage maps, etc. However, the mobile crowdsourcing system which is being used currently have some genuine problems, like motivating the workers to take interest in participation and do well while performing task. This can be solved by requester if he/she reward the workers whose work output is valuable and correct.

One of the most mainstream interests for the mobile crowdsourcing systems is how can we increase the social welfare. The main focus in the research of upgrading the system, being the foundation of incentive mechanisms. Traditional incentive mechanisms use two types of settings namely offline and online settings. The incentive mechanism design gets complicated due to the mobile nature of these circulated computation and sensing powers. In the mobile crowdsourcing system the requester post the task together with related reward budget. The interested worker uploads his bidding, also known as the solution of the task (comprised of detecting cost and time) to the requester. After analyzing the bidding the requester decides whether to accept or reject this worker.

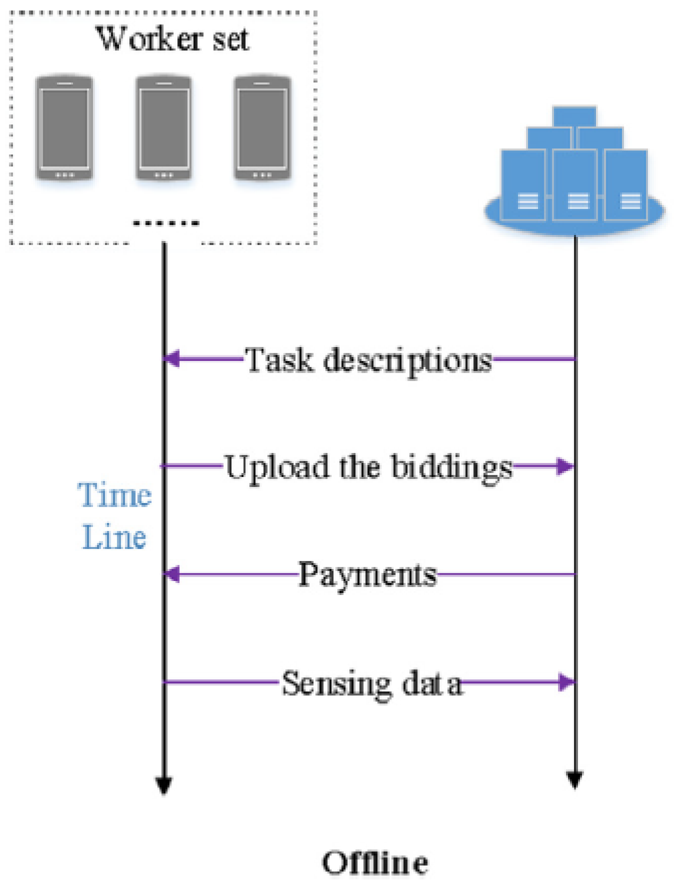


Fig 2.1 – Traditional offline settings for incentive mechanism

The working of offline incentive mechanisms (Fig 2.1) consist of deciding the winners present in the auction after every participants have uploaded their bidding. These offline mechanism expect that every worker should be present from the first round itself of task distribution and it can’t accept new workers or bidding afterwards. We can use the online incentive mechanisms (Fig 2.2) to resolve the issues of offline incentive mechanisms. However, even online incentive mechanism has its own set of disadvantages, which we will be discussing after this.

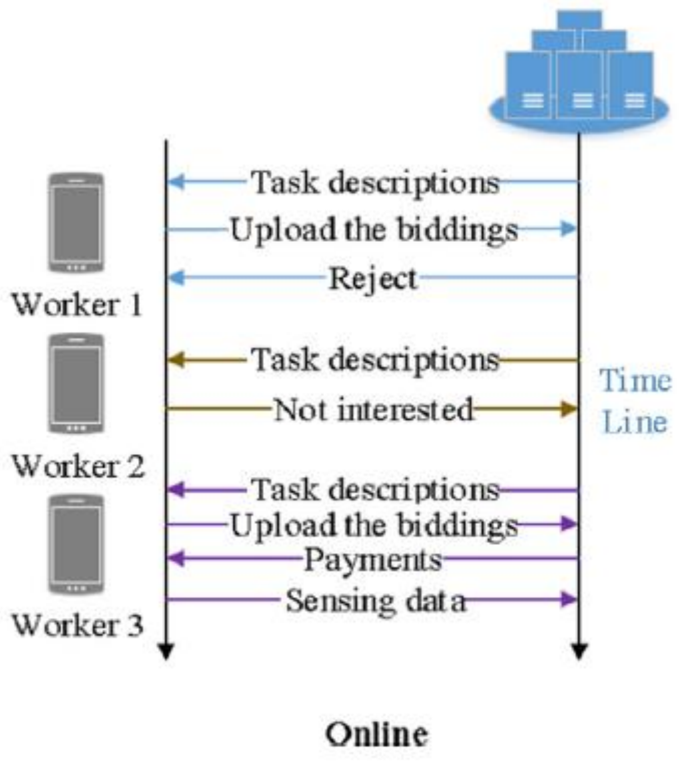


Fig 2.2 – Traditional online settings for incentive mechanism

However, the online incentive mechanism sometimes bring in an inefficiency in the process of auction when it fails to successfully choose the correct arrangements of candidates from the reputation database created for the workers. Therefore, to design a new incentive mechanism we can combine offline and online settings. (Fig 2.3).

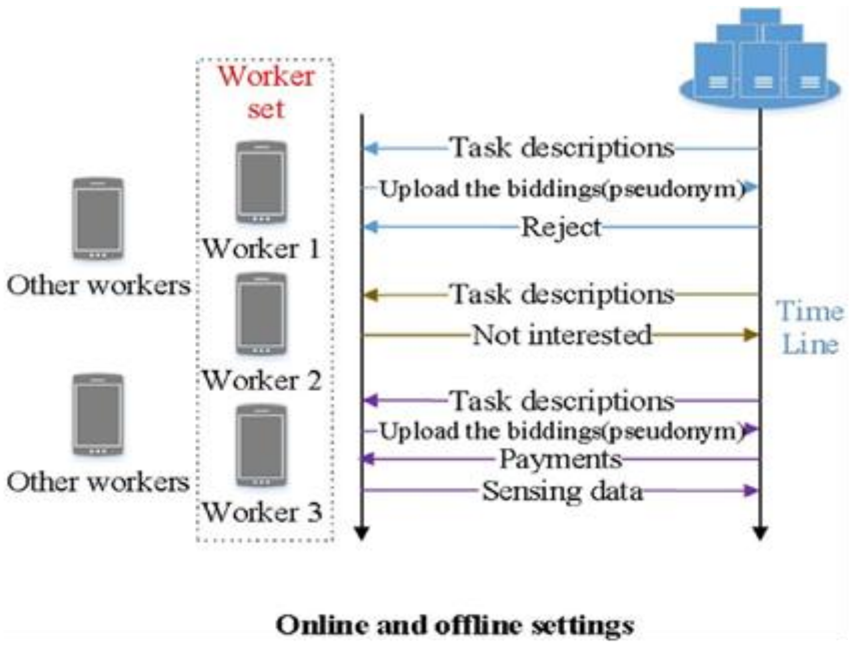


Fig 2.3 – The processing procedure for the proposed incentive mechanism

In the devised incentive mechanisms, the worker set that can be appointed to complete the given task is decided by the platform based on the offline incentive mechanisms. After that, once the worker are selected and arrived the transaction between workers and platforms are done based on the online incentive mechanism. Also, the privacy protection is included for the workers who all have participated. Therefore, the devised incentive mechanism advantages are that, it can overcome the disadvantages of offline and online incentive mechanisms, provide privacy and security to workers' data and increase the productivity of the mobile crowdsourcing system.

2.1 ESTABLISHING EFFECTIVE INCENTIVE MECHANISM

Setting up effective incentive mechanisms is a challenging research focus since, privacy assurance and selfishness are the major topic to be taken into consideration. We will discuss the aspects on which the mobile crowdsourcing mostly works.

2.1.1 On the aspect of incentive mechanisms

There are two normal plan namely, offline settings and online settings. In offline incentive mechanism, which utilize Stackelberg, game. In this mechanism platform is the leader and the user are the adherents. There are two system models described as: first, platform-driven model where the reward is given by the platform which is shared by the participating users and second, the user-driven model where the payment which is to be received is controlled by users itself. The working of the offline incentive mechanisms is depicted in the Fig 2.1.

However, the disadvantage of offline incentive mechanisms is that it expects the users to join the bidding right from the start of the session and can't accept new biddings later on. The online incentive mechanisms is based on online reverse auction are provided, Truthful Online Incentive Mechanisms (TOIM) and Threshold Based Auction (TBA). The disadvantage of online incentive mechanisms is that it fails to select the workers from the workers' reputation database, which results in inefficiency during the whole process of the auction.

2.1.2 On the aspect of auction algorithms

In this aspect the more general approach is two-stage auction algorithm. The process of two-stage auction algorithm consist of rejecting and utilizing the first batch of users as the sample, which will help in making a detailed choice on whether we can accept the rest of users. But, disadvantage of this strategy is that it fails to guarantee the power of consumer, since the first batch of user can never win no matter how low their cost is. To encourage the users to arrive late, this method consequently rejects the first batch of users. This shows that the users who arrive early have no impetus to send in their biddings, which will result in ruining the users' competition and even result in task starvation.

An auction-based incentive mechanism uses the announced total reward R (budget) and user i's sensing plan t_i (Ability of user i that for how long a duration he can be in the sensing task). The requester's target which is under a budget increases. This strategy is known as spending achievability, and the mechanism is composed such that the sum of its payments does not surpass the budget. Thus, the two critical components that is budget and sensing plan for developing an efficient auction algorithm in crowdsourcing systems, for example, Amazon Mechanical Turk, a budget oriented successful crowdsourcing system. However, in the above mechanisms the auction limits cannot be changed dynamically and are static.

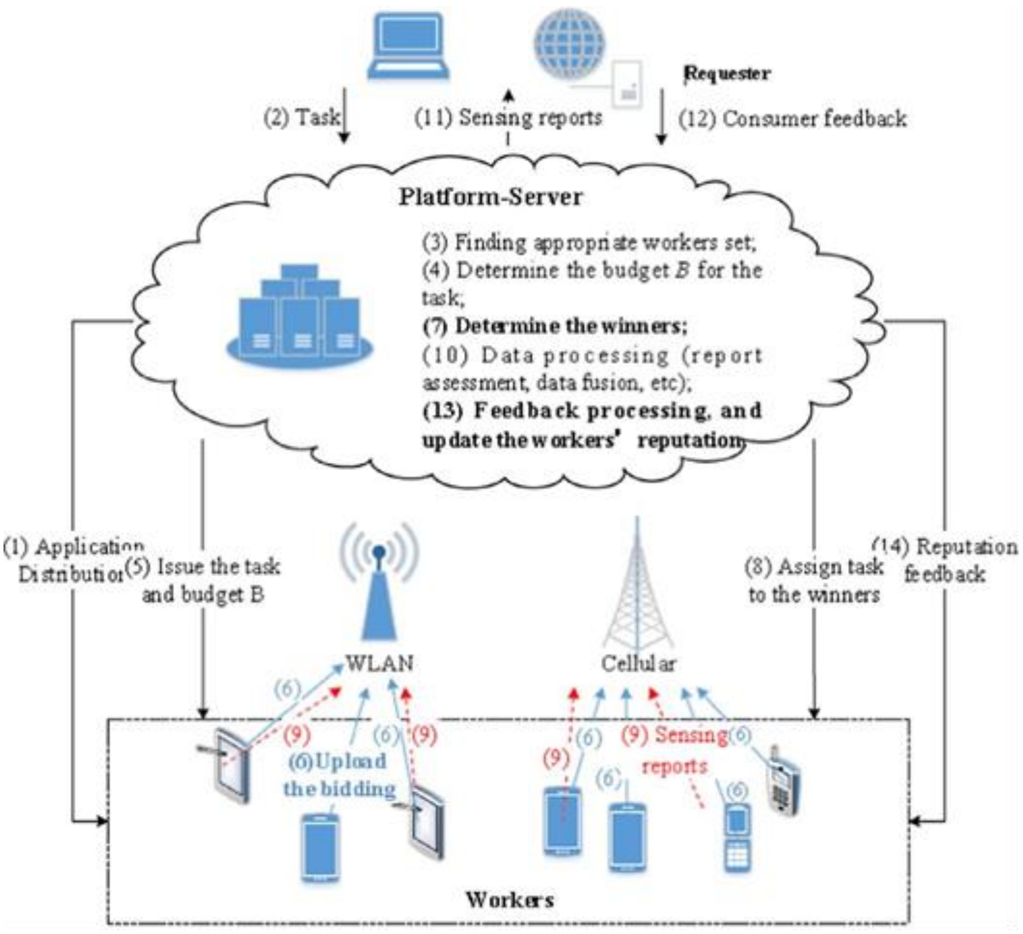

Requester
(2) Task
(11) Sensing reports
(12) Consumer feedback
Platform-Server
(3) Finding appropriate workers set;
(4) Determine the budget B for the task;
(7) Determine the winners;
(10) Data processing (report assessment, data fusion, etc);
(13) Feedback processing, and update the workers' reputation
(1) Application Distribution
(5) Issue the task and budget B
(8) Assign task to the winners
(14) Reputation feedback
WLAN
Cellular
(6)
(9)
(6)Upload the bidding
(9) Sensing reports
Workers

Fig 2.4 – The structure of a mobile crowdsourcing system

2.1.3 On the aspect of incentive and punishment strategies

Many incentive and punishment strategies were proposed to solve the problem of free-riding problem. A whole lot of the incentive mechanisms rely on fiscal prize present in the form of micropayments. The workers are paid in cash upon the completion of a task. The social dilemma existing between the workers and platform go unsolved and is not understood by the current pricing schemes. Also, due to the unique threshold in this incentive mechanism, some good and potential workers go unnoticed and thus creating an unfair problem.

2.2 PROPOSED MECHANISM

Following are the mechanism, which we will be implementing to diminish the common challenges in mobile crowdsourcing:

1. We will be combining the advantages of offline incentive mechanism and online incentive mechanisms to develop an incentive mechanism that picks the worker statically and the then the winner are chosen progressively after bidding. For this

incentive mechanism framework the protection of the privacy of the worker is kept in mind.

2. We will be implementing an Improved Two-stage Auction algorithm (ITA) by the help of which we can decide the winners in real-time for the given platform. This will also solve the problems of workers like injustice and unfairness problem and will also encourage the workers to arrive early.
3. To confirm the adequacy of our used algorithm or incentive mechanism, we compare it with some already used incentive mechanisms and algorithms and run through simulation. The results will contain the advantages and improvements of our devised algorithm.

2.3 EXISTING CROWDSOURCING APPLICATIONS

In this section, we will discuss about some applications, which uses crowdsourcing method.

2.3.1 Predicting bus arrival time in Singapore

People using bus as their mode of transportation always want to know the exact arrival time of the bus. Long waits at the bus stops often make reluctant to take buses. Online bus schedules provided by bus operating companies are not timely updated and provide limited information. Therefore, there should be some mechanism to predict bus location and its arrival time. Crowd participated design i.e. bus arrival time prediction system was designed to satisfy above requirement.

Querying Users - Users having mobile phones, querying the bus arrival time for a particular bus route.

Sharing Users - Users having commodity mobile phones or sensors to sense lightweight cellular signals and surrounding to backend server.

Backend Server - It collects information from sharing users and process it to monitor bus routes and predicts bus arrival time.

Therefore, this system avoids the use of GPS, as it consumes large amount of energy in terms of battery life.

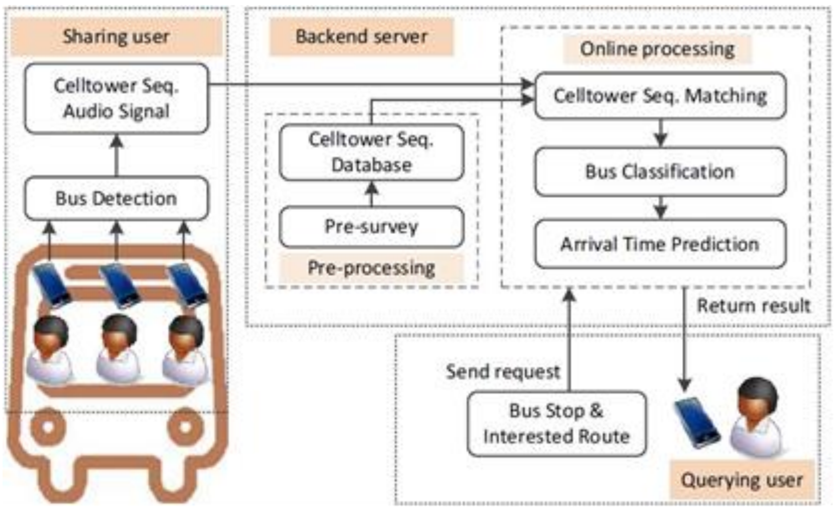


Fig 2.5 – Predicting Bus Arrival Time System Architecture

The mobile phone of a person who is on the bus detects the beep audio response of the card reader that is used for payment of bus fees. This helps in detecting that whether the person is on the bus or not. But similar sound is heard in other kind of transport like in Rapid Train system (MRT in Singapore). But accelerometer can be used to differentiate between the two. As the variance in acceleration of the bus is more than that of trains, it can be detected that person is on bus or train.

The backend server maintains a database of cell tower IDs that come in different bus routes. Sharing users present in bus sense the environment and report the sensed cell tower IDs to server. At a time, it can be connected to multiple cell towers with varied signals strength. So to classify the bus route, sequence matching is performed for top-3 strongest signal strength cell towers. For matching purpose, they use Smith Waterman algorithm which is a dynamic programing algorithm and performs local sequence alignment.

If the length of the sequence uploaded is below some threshold, then Cell-tower sequence of several users on the same bus are concatenated to form longer length sequence. To identify if users are on same bus or not, time difference between different beep signals from IC card readers is captured. After applying sequence matching algorithm, backend server estimates the arrival time according to both historical data as well as latest bus route information.

Advantages

- This system is power efficient as it is independent of use of GPS.
- No explicit input is required from user. Automatic detection of beep sound and signals.
- Processing and computation part performed on backend server, which will not affect the performance of user mobile phones.

Disadvantages

- If no users are present in the bus, cell tower IDs are unable to trace.
- Arrival time of the first few bus stops will not be timely updated as it takes time to capture the signals and meet threshold length for bus route classification.

2.3.2 mClerk

It is a paid mobile crowdsourcing platform used for digitizing local language documents. It send and receive tasks through SMS to make it accessible to anyone using low-end mobile phone in developing regions. Small bit mapped images for digitization are sent via ordinary SMS so that worker can get and reply messages without any internet connection. For digitization purpose, workers receive an image of a word to be typed and send back the typed version of the same via SMS only.

- mClerk system scans the paper document first, then decomposes the document into images of words, sends each image to users' phone, receives the users' responses, verifies them by duplicating, pays workers and combine responses into a digital document.
- **Image Segmentation**: Currently paper documents containing textual data are used as input. Scanned pages of handwritten or printed text are segmented into separate words.
- **Mobile Crowdsourcing**: Decomposed images are sent to workers' mobile phones and they send back the typed version as response. Most of the mobile phones lack local language font support e.g. typing in Hindi can be tedious and error prone (for vowels). But for this system, users with basic knowledge of English, sends best English transliteration of a word.
- **Verification of responses**: This is done by sending same image to multiple users and comparing their responses. As users send transliterations, it is possible that system might reject distinct responses although they are similar. So Verification

algorithm considered two responses equivalent if both transliterate to same word in local language.

Payment Mechanism: Mobile airtime in chunks of INR 10 are given to users for providing correct responses, as it does not require any bank account.

Users and Referrals: To increase the individual throughput and total user count, user could refer and register his friend by giving missed call to same number from which he receives task. Researcher calls back and registers new user. For this referral mechanism, users are paid 10 percent of total earnings of their referrals. Missed call system was also used in case users had queries.

Providing feedback and motivational messages to users: Users were paid when the total amount earned from messages and referrals reached INR 10. Also, if user did not reply for more than 24 hours, system sends reminder message to user to urge them to reply again. At the end of the day, leaderboard messages displayed names and earnings of top 5 workers so as to motivate users and increase their participation level.

This system achieved accuracy of 90.1%.

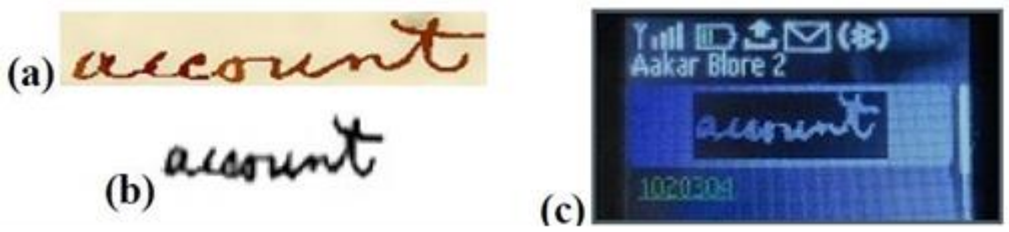


Fig 2.6 – (a) Original image containing the word (b) Same word in SM format (c) Sample of the SM message

Advantage

- Uses SMS based approach, which enables the participation of more users. As no need to login repeatedly in case of web based approach.
- Gives freedom to users to choose time and duration for which they want to work.
- Compensation in terms of mobile balance helps in motivating users.

Disadvantage

- This system does not maintain reputation of user so as to ensure quality of response.
- English to local language converter can give incorrect transliteration although user gave correct response.
- Although users will enter a word but it could be a wrong English transliteration.

CHAPTER 3
SYSTEM DEVELOPMENT

3.1 SYSTEM OBJECTIVE

Our main objective is to build an online incentive mechanism which will have the following five properties as stated:

1. *Computational efficiency*. To make this mechanism computationally effective, the time complexity should be restricted to polynomial.
2. *Individual rationality*. A non-negative utility will be given to user after he will complete the given sensing task to him/her.
3. *Profitability*. The platform will get non-negative utility towards the end of a sensing task.
4. *Fault Tolerance.* To tolerate any fault that could arise due to lost in connectivity of database.
5. *Truthfulness*. A mechanism is can be termed as truthful, if a worker can't enhance his utility by submitting a price of bidding straying from his true value in spite of other worker's bidding prices.

3.2 DESIGN AND IMPLEMENTATION

While planning this system, we combine the offline and online incentive mechanisms to build an incentive mechanism that will select the workers statically and afterwards the winners are selected progressively after bidding. Let the set of workers be C = (1, 2, …, n), where n defines the total number of workers. Due to the dynamic auction process we have to consider "zero arrival-departure" in this design. In this system there are three cases that completes a single transaction in between the platform and worker:

1. In this stage the workers are allotted the task and budget (B). The workers in return uploads their corresponding bidding ($b_{i)}$ and sensing plan (t_i). The worker is rejected by the platform because of discontent.
2. In this stage the worker *i* who isn't interested in the task is allotted the task and budget (B).
3. In this stage the worker who uploads his/her bidding b_i and sensing time t_i is allotted the task and budget (B). This stage understand the potential of the worker *i* and transfers the payment to worker who in return handover the task report.

Due to the rejecting nature of the system the workers will be inspired to submit correct and proper values of the required parameters. This will encourage worker to perform well to get rewards and be a winner.

3.3 DEVELOPMENT

In this system a set of task is defined, for the workers to choose, denoted by $\phi = (\phi 1, \phi 2, ..., \phi m)$. According to the task selected every worker has two things to the platform which is private and is not known to other workers. They are contribution value $v_i > 0$ and associated cost c_i. When the worker i is bidding, his/her bidding is represented by the service price of the worker that he will charge for the task completion and is denoted by b_i. Also, there is a parameter *sensing plan*, which is the time duration the worker can provide the service and is denoted by t_i. The worker submits his/her parameters to the platform i.e. b_i and t_i. After receiving all the information by the workers the platform decides the winner workers and store it in winner set W and sets the payments for each winning worker denoted by p_i. Eq. 1 depicts this:

$$u_i = \begin{cases} p_i - c_i, i \in W \\ 0, Otherwise \end{cases} \tag{1}$$

We define $c_i = \tau \times t_i$, where unit cost of workers are denoted by τ and $0 < \tau < 1$. But, in Eq. 1, parameters which determine c_i are t_i and τ, but, with the change of sensing time t_i at the end, c_i will change too. Therefore the "real utility of worker i" at the end is shown by Eq 2.

$$u_i' = \begin{cases} p_i - c_i', i \in W \\ 0, Otherwise \end{cases} \tag{2}$$

In addition, t_i and budget B determines p_i, where the "budget for a specific task" is represented by B and the platform determines it. Shown by Eq. (3).

$$p_i = \frac{t_i}{T} \times B \tag{3}$$

Where "maximal sensing time" for this task is T, and defining $B/T \geq 1$. Eq. 4 defines the utility of the platform.

$$\bar{u} = \lambda \times \log\left(1 + \frac{v(w)}{\lambda}\right) - P(W) \tag{4}$$

The total benefit of the problem is denoted by $V(W) = \sum_{i \in W} v_i$, and total payments for the selected worker is denoted by $P(W) = \sum_{i \in W} p_i$. To control the gradient of the return, λ is used as a system parameter, and $\lambda > 1$.

3.4 ALGORITHM

3.4.1 Improved Two-stage Auction algorithm

In Fig 2.4, step (6) shows the uploading the biding and step (7) is for determining the winners in improved two-stage auction algorithm. The improved two-stage auction algorithm which helps in determining the winner workers in real-time for the system, is shown in the step (6) and (7). This auction is divided in two parts or stages. In the first stage we collect the details and build a base for the upcoming stage. It is different from the previous answers, in this stage the workers coming in the first batch can also win the bidding and thus solving the unfairness problem. Worker's will also be encouraged by this method to arrive on time. We adjust the bidding threshold dynamically, in transaction in the second stage. It is done on the basis of result collected form the first stage. The process is shown as follows:

1. The budget B and the maximum sensing time T is specified by the system itself. Based on the decided parameter and experience from the history, the threshold κ is decided by the platform for the process of bidding.
2. The marginal budget *B'* based on T and B is determined by platform for stage 1. Let the payment sum be $P' = \sum_{j \in \Gamma} p_j$ in stage 1. The winner set is represented by Γ in the stage 1. Also, the value if *B'* is determined by Eq. 5. This also dynamically determines the sample size and expresses the value of *B'* based on "multiple-stage sampling-accepting process":

$$B' = \left\lfloor \frac{B}{2^{\lfloor \ln T \rfloor}} \right\rfloor \tag{5}$$

Also, the limited time in stage 1 is denoted by *T'*. So, *T'* is derived by using Eq. 6

$$T' = \left\lfloor \frac{T}{2^{\lfloor \ln T \rfloor}} \right\rfloor \tag{6}$$

ITA solves the main problem of unfairness and workers arriving early. The problem were present in traditional two-stage auction algorithm. ITA also improves the efficiency of the auction.

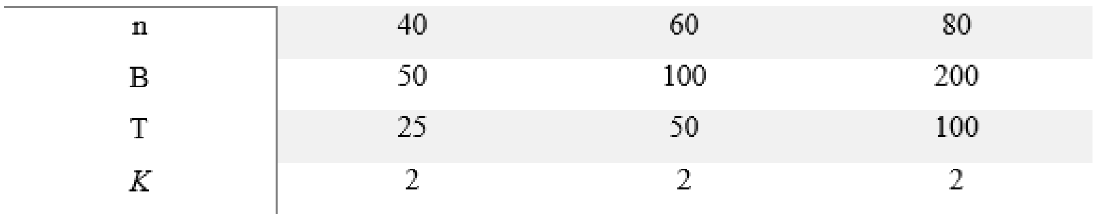
CHAPTER 4
PERFORMANCE ANALYSIS

We conduct reenactments to assess the proposed incentive mechanism. We check the effectiveness of ITA, through contrasting it with other auction algorithms, which are more general.

4.1 EFFICIENCY OF ITA

To compare ITA with other auction algorithm we will auction algorithm. To achieve this for different situation we take three set of testing having budgets and sensing time different for each case. 50, 100 and 200 will be the different budget for the testing. Simultaneously, the required aggregate sensing times will be set to 25, 50 and 100 respectively. The total number of workers are 40, 60 and 80 respectively. For this set of experiment we set threshold κ = 2 based on the skills of works. All the parameters are shown in table 4.1. In this experiment, we run the improved two-stage auction algorithm against two-stage auction algorithm and general auction algorithm.

Table 4.1 – The setting of parameters for ITA



n	40	60	80
B	50	100	200
T	25	50	100
K	2	2	2

To compare the algorithms and their efficiency we record the aggregate payment for the worker: *P(W)* in these recreations. The x-axis is for “transaction time sequence” and y-axis if for the “values of *P(W)*”. Under the different test cases and budget the proposed algorithm gives good efficiency over other algorithm. Thus after comparing the *P(W)* we can give the reasons as to why ITA is preferred over other algorithms.

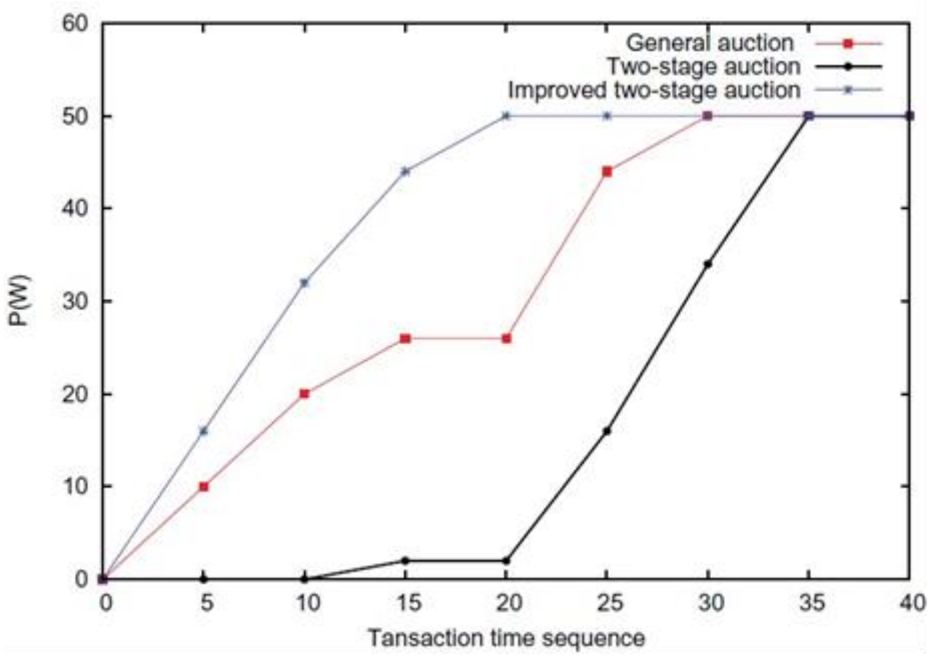


Fig 4.1 - Setting budget of task as 50 to compare auction algorithm.

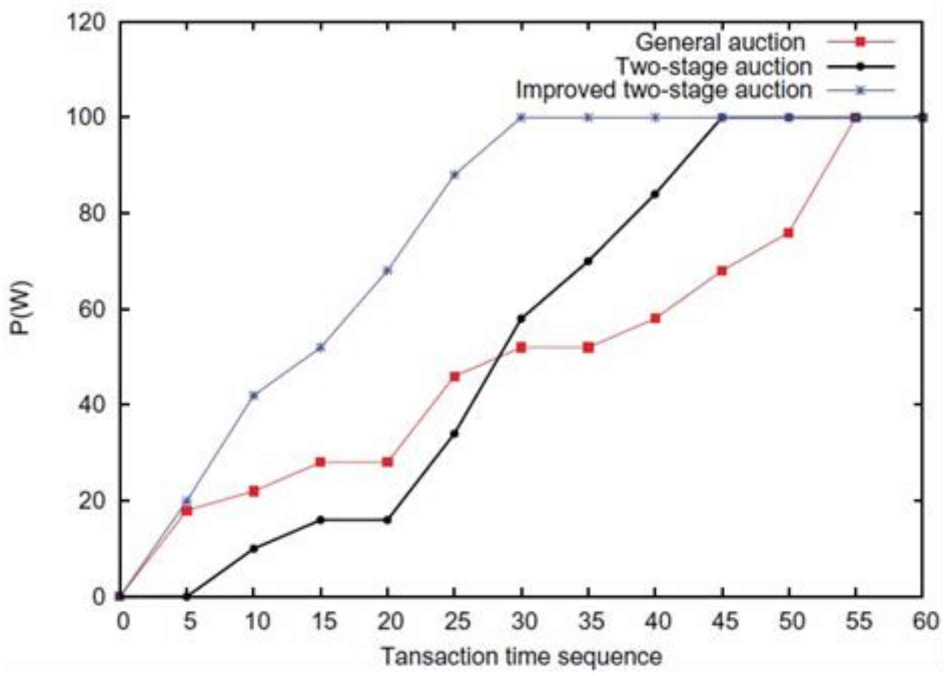


Fig 4.2 - Setting budget of task as 100 to compare auction algorithm.

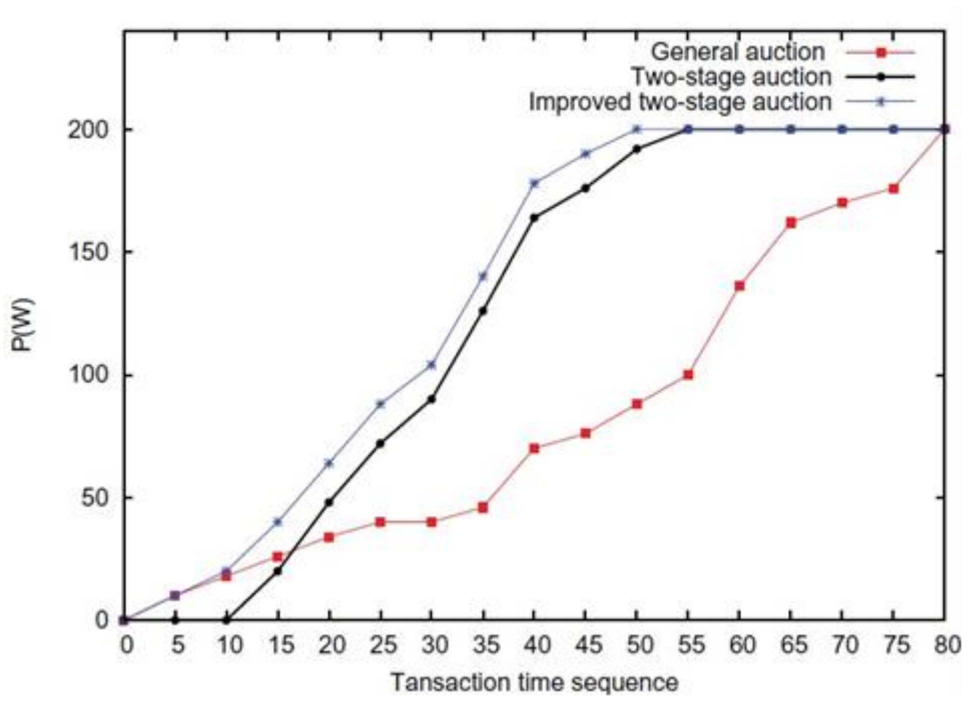


Fig 4.3 - Setting budget of task as 200 to compare auction algorithm.

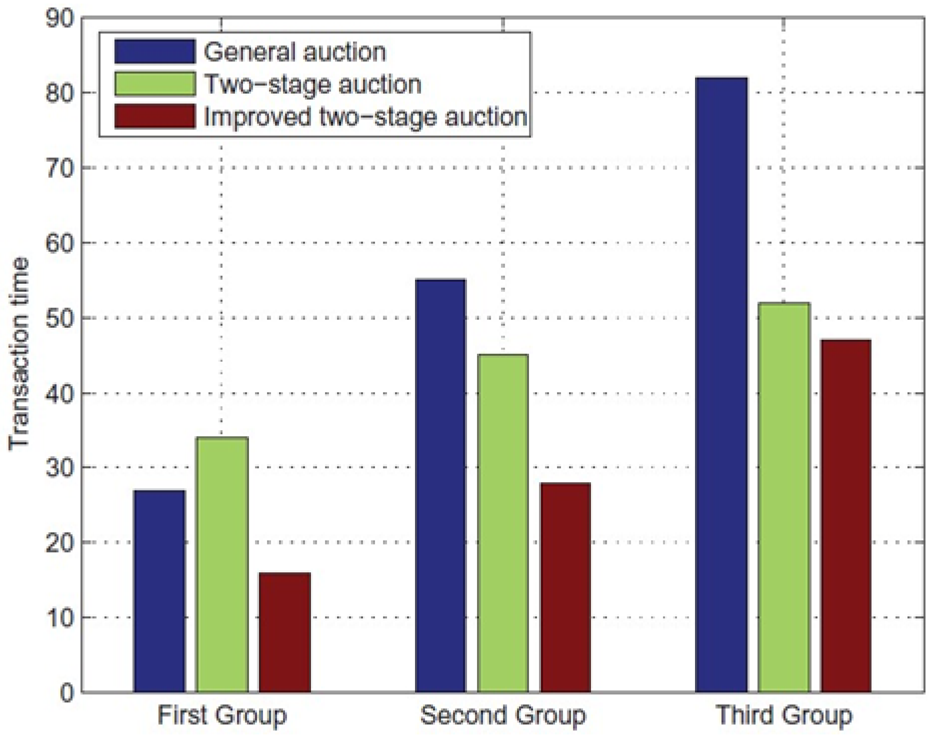
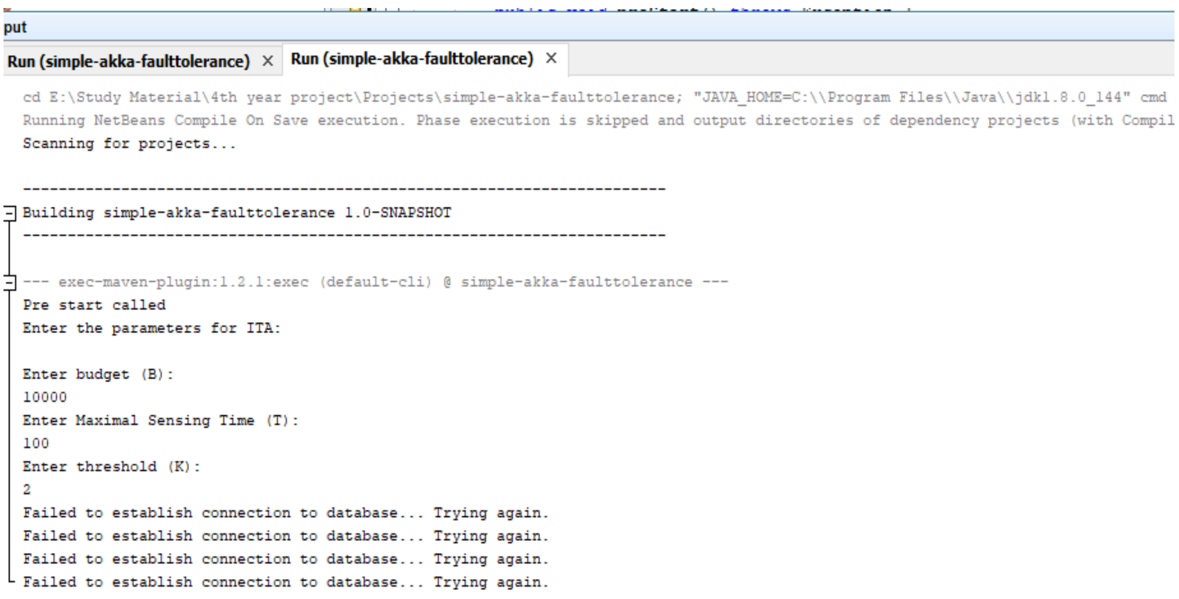


Fig 4.4 - The comparative results of time consumptions.

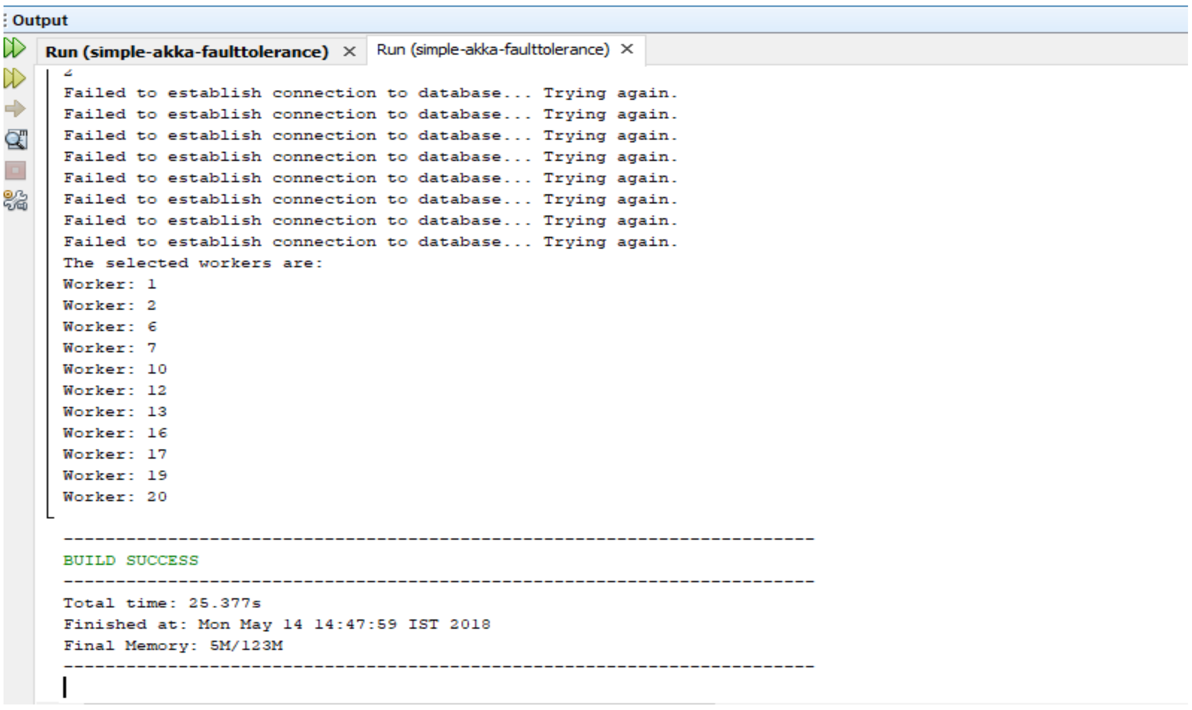
4.2 OUTPUT OF THE CODE IMPLEMENTED


put
Run (simple-akka-faulttolerance)
Run (simple-akka-faulttolerance)

```
cd E:\Study Material\4th year project\Projects\simple-akka-faulttolerance; "JAVA_HOME=C:\\Program Files\\Java\\jdk1.8.0_144" cmd
Running NetBeans Compile On Save execution. Phase execution is skipped and output directories of dependency projects (with Compil
Scanning for projects...

------------------------------------------------------------------------
Building simple-akka-faulttolerance 1.0-SNAPSHOT
------------------------------------------------------------------------

--- exec-maven-plugin:1.2.1:exec (default-cli) @ simple-akka-faulttolerance ---
Pre start called
Enter the parameters for ITA:

Enter budget (B):
10000
Enter Maximal Sensing Time (T):
100
Enter threshold (K):
2
Failed to establish connection to database... Trying again.
Failed to establish connection to database... Trying again.
Failed to establish connection to database... Trying again.
Failed to establish connection to database... Trying again.
```

Fig 4.5 – Connection is not established

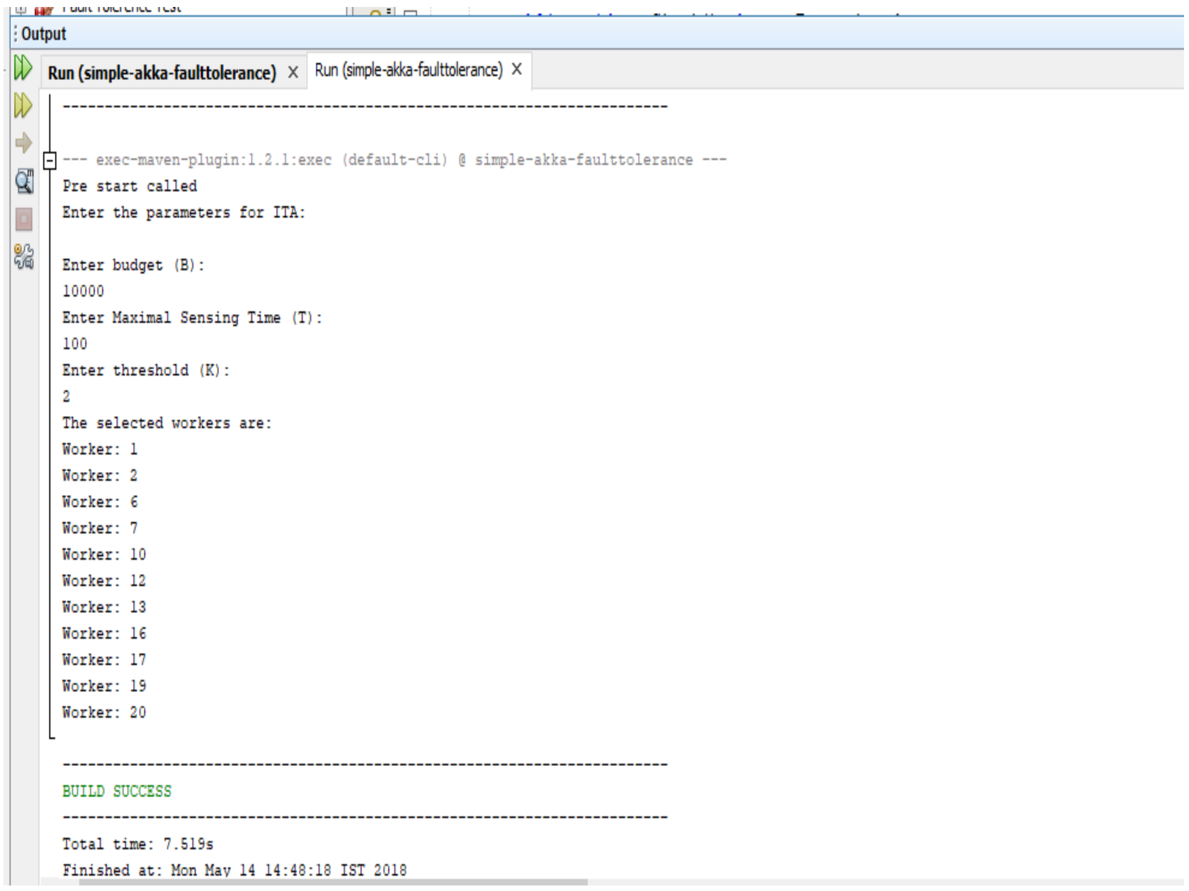
The output in Fig 4.5 is depicting how the program is behaving when it tries to connect to database but the connection is not established because of some error at the database end or the error in network. The actor in the program keeps on trying to connect to database with a limit of 500 attempts. Since the connection of database and other working of program is on different threads, the program need not to exit with an error.


Output
Run (simple-akka-faulttolerance)
Run (simple-akka-faulttolerance)

```
Failed to establish connection to database... Trying again.
Failed to establish connection to database... Trying again.
Failed to establish connection to database... Trying again.
Failed to establish connection to database... Trying again.
Failed to establish connection to database... Trying again.
Failed to establish connection to database... Trying again.
Failed to establish connection to database... Trying again.
Failed to establish connection to database... Trying again.
The selected workers are:
Worker: 1
Worker: 2
Worker: 6
Worker: 7
Worker: 10
Worker: 12
Worker: 13
Worker: 16
Worker: 17
Worker: 19
Worker: 20

------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time: 25.377s
Finished at: Mon May 14 14:47:59 IST 2018
Final Memory: 5M/123M
------------------------------------------------------------------------
```

Fig 4.6 – Connection was established while the actor was still trying to connect

Since, the actor is running on the different thread it will try to connect to database irrespective of the other threads. As soon as the connection is established, the actor will reconnect with the database, as seen in the Fig 4.6, and will carry out the result.



Output

Run (simple-akka-faulttolerance) × Run (simple-akka-faulttolerance) ×

```
------------------------------------------------------------------------

--- exec-maven-plugin:1.2.1:exec (default-cli) @ simple-akka-faulttolerance ---
Pre start called
Enter the parameters for ITA:

Enter budget (B):
10000
Enter Maximal Sensing Time (T):
100
Enter threshold (K):
2
The selected workers are:
Worker: 1
Worker: 2
Worker: 6
Worker: 7
Worker: 10
Worker: 12
Worker: 13
Worker: 16
Worker: 17
Worker: 19
Worker: 20

------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time: 7.519s
Finished at: Mon May 14 14:48:18 IST 2018
```

Fig 4.7 – No error in connection and program execute in the first attempt

If the connection was established at the starting of the program, the program will run as intended and output will be printed in the first attempt, as shown in Fig 4.7.

CHAPTER 5
CONCLUSIONS

5.1 CONCLUSION

After implementing the ITA algorithm and fault tolerance akka API and running it through simulation for three different groups of parameters on MATLAB, we can see for every case tested the ITA is the fastest algorithm compared to any other auction algorithm. It can complete the task faster compared to others and the fault in connecting database was handled by the program too. In last, we can conclude that improved two-stage auction algorithm performs any task on the platform faster than others do.

5.2 FUTURE SCOPE

For the future works, we can focus on the behavior of workers for malicious fluctuation. One of the malicious fluctuating behavior is when a worker gathers good reputations in previous tasks and then behave unreliably in future tasks. In addition to it, we will tolerate more faults like no table found, no database found, etc. We will also up the security measures in the database by applying encryption method.

REFERENCES

[1] Yingjie Wang, Zhipeng Cai, Guisheng Yin, et al. “An incentive mechanism with privacy protection in mobile crowdsourcing systems”, “Computer Networks”, 2016.

[2] Kan Yang, Kuan Zhang, Ju Ren, et al. “Security and Privacy in Mobile Crowdsourcing Networks: Challenges and Opportunities”, “IEEE Commun. Mag”, Vol 53 issue 8, 2015.

[3] R. K. Ganti et al., “Mobile Crowdsensing: Current State and Future Challenges”, “IEEE Commun. Mag.”, vol. 49 no. 11, Nov. 2011.

[4] D. Christin et al., “A Survey on Privacy in Mobile Participatory Sensing Applications,” “J. Systems and Software”, vol. 84 no. 11, 2011.

[5] Y. Wang, Y. Huang, and C. Louis, “Respecting user Privacy in Mobile Crowdsourcing”, “Science”, vol. 2 no. 2, 2013.

[6] L. Zhang, Z. Cai, J. Lu, et al, “Mobility-aware routing in delay tolerant networks”, “Pers. Ubiquit. Comput”, vol. 19, 2015.

[7] X. Wang, Y. Lin, S. Zhang, et al, “A social activity and physical contact based routing algorithm in mobile opportunistic networks for emergency response of sudden disasters”, “Enterp. Inf. Syst.”, vol. 3, 2015.

[8] D. Zhao, X. Li, H. Ma, “How to crowdsource tasks truthfully without sacrificing utility: online incentive mechanism with budget constraint”, “The 33rd IEEE Conference on Computer Communications, 2014.

[9] Erin Brady, Meredith Ringel Morris, and Jerey P Bigham, “Gauging receptiveness to social microvolunteering”. “In Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems”, ACM, 2015.

[10] Ju Ren, Yaoxue Zhang, Kuan Zhang, et al, “Exploiting mobile crowdsourcing for pervasive cloud services: challenges and solutions.”, “Communications Magazine”, IEEE, 2015.

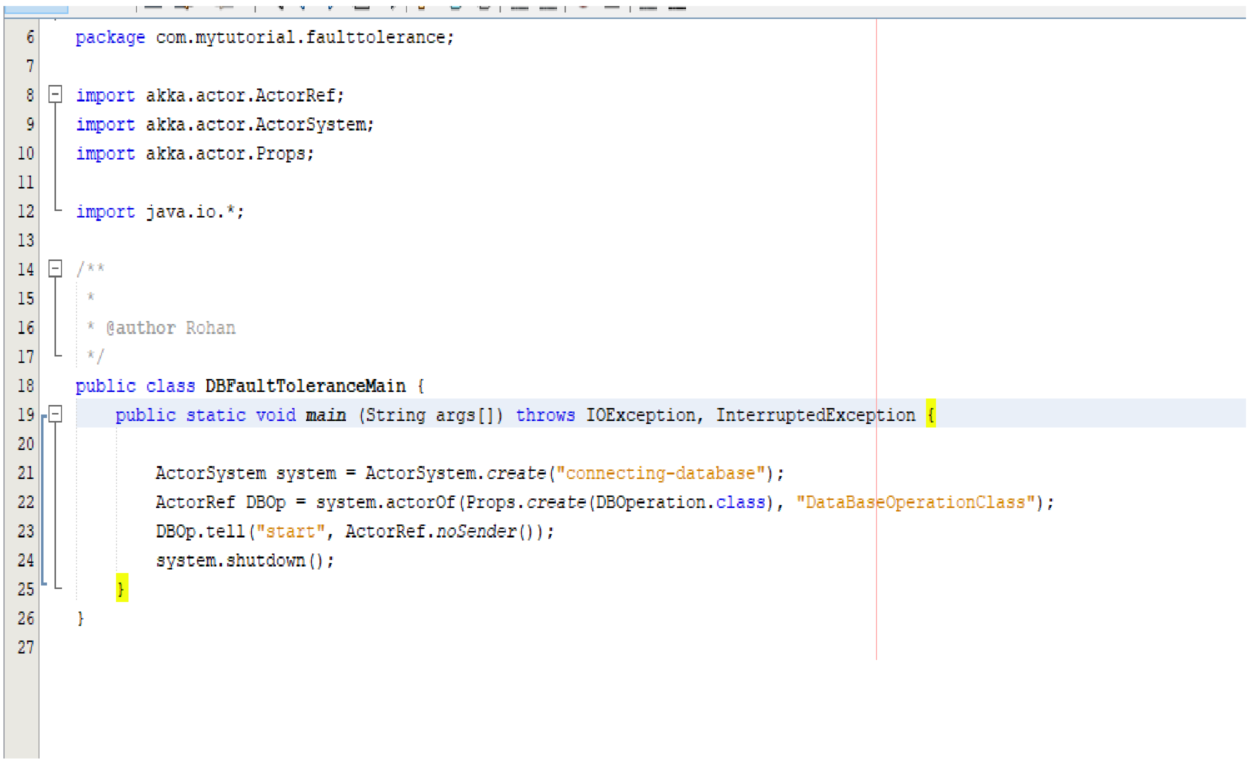
[11] John P Rula, Vishnu Navda, Fabi_an E Bustamante, et al, “No one-size fits all: Towards a principled approach for incentives in mobile crowdsourcing.”, “In Proceedings of the 15th Workshop on Mobile Computing Systems and Applications”, ACM, 2014.

[12] Pengfei Zhou, Yuanqing Zheng, and Mo Li, “How long to wait?: predicting bus arrival time with mobile phone based participatory sensing.”, “In Proceedings of the 10th international conference on Mobile systems, applications, and services”, ACM, 2012.

APPENDICES

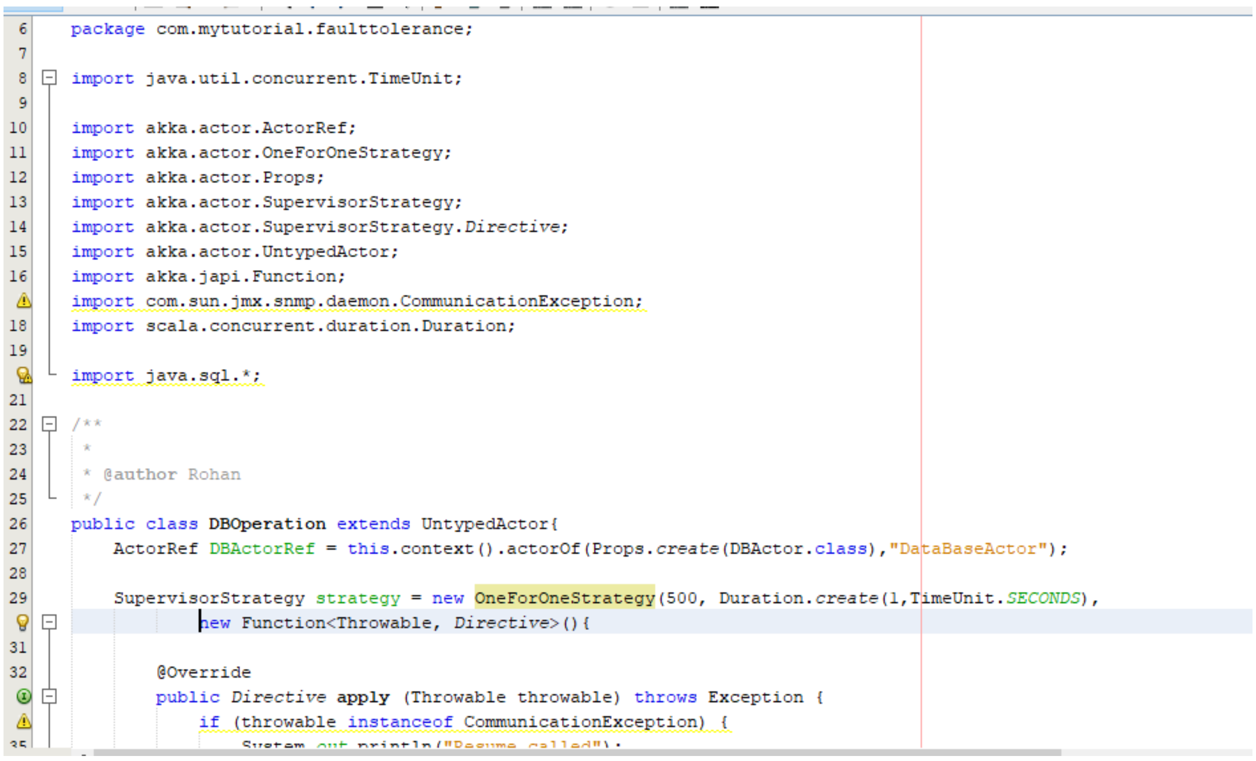
Implemented code is listed below-

1. DBFaultToleranceMain.java

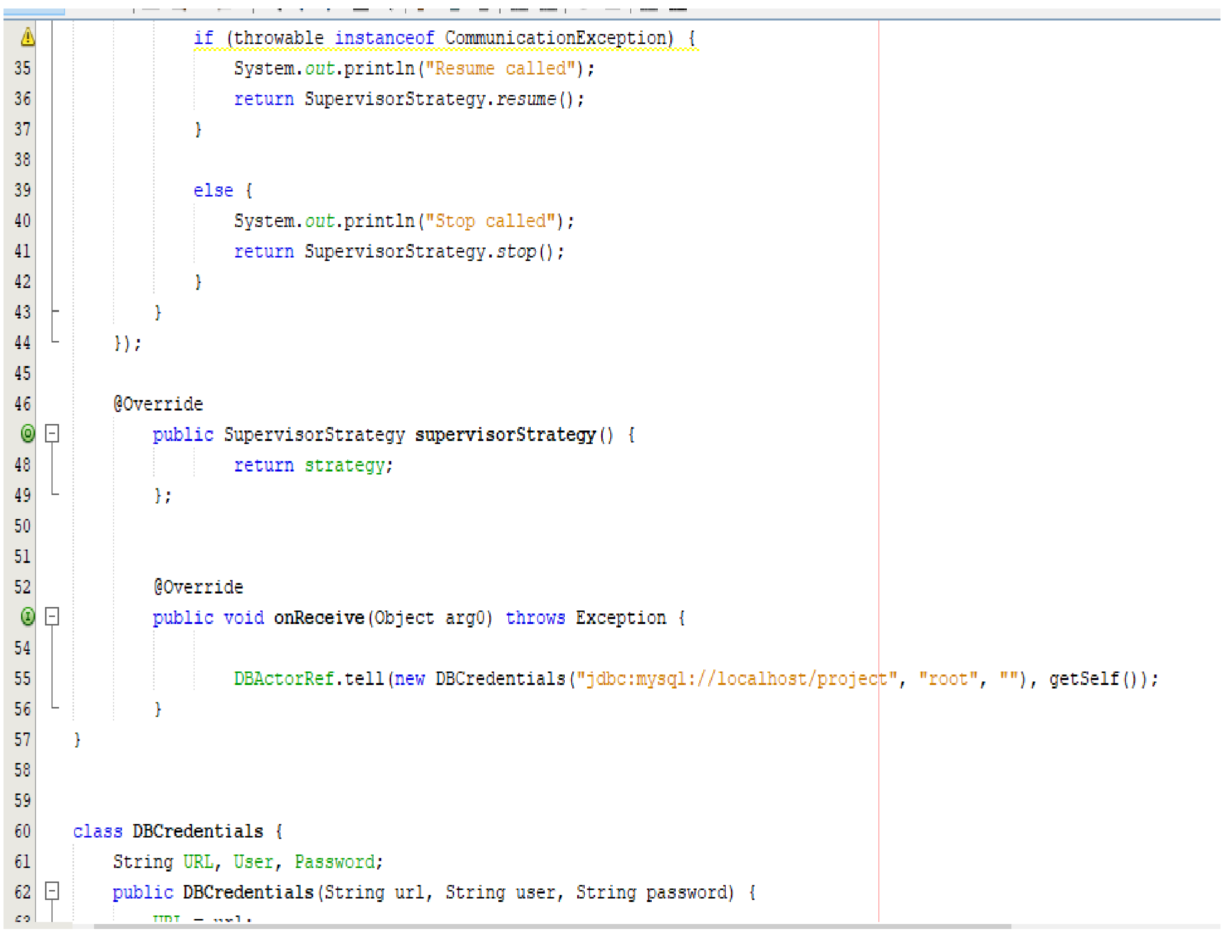


```java
package com.mytutorial.faulttolerance;

import akka.actor.ActorRef;
import akka.actor.ActorSystem;
import akka.actor.Props;

import java.io.*;

/**
 *
 * @author Rohan
 */
public class DBFaultToleranceMain {
    public static void main (String args[]) throws IOException, InterruptedException {

        ActorSystem system = ActorSystem.create("connecting-database");
        ActorRef DBOp = system.actorOf(Props.create(DBOperation.class), "DataBaseOperationClass");
        DBOp.tell("start", ActorRef.noSender());
        system.shutdown();
    }
}
```

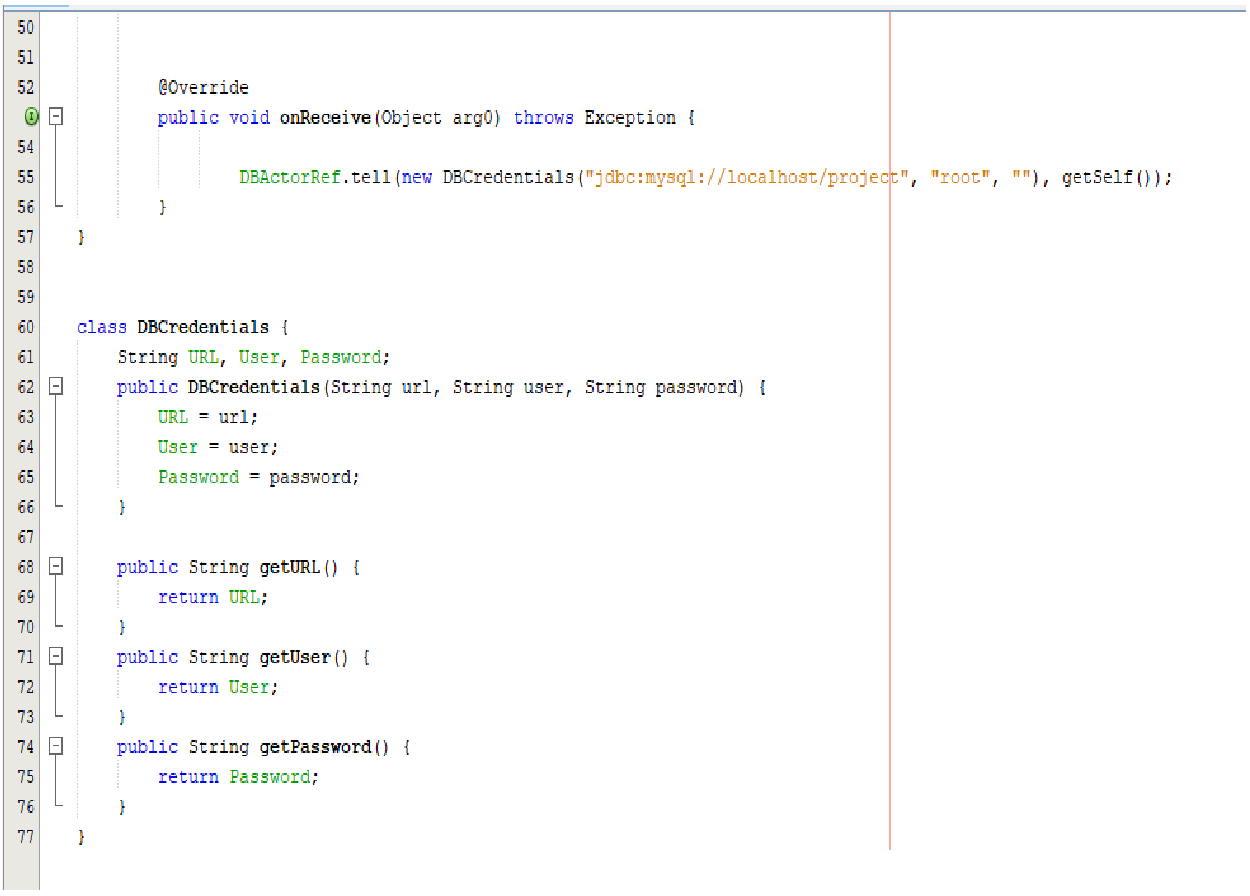
2. DBOperation.java



```java
package com.mytutorial.faulttolerance;

import java.util.concurrent.TimeUnit;

import akka.actor.ActorRef;
import akka.actor.OneForOneStrategy;
import akka.actor.Props;
import akka.actor.SupervisorStrategy;
import akka.actor.SupervisorStrategy.Directive;
import akka.actor.UntypedActor;
import akka.japi.Function;
import com.sun.jmx.snmp.daemon.CommunicationException;
import scala.concurrent.duration.Duration;

import java.sql.*;

/**
 *
 * @author Rohan
 */
public class DBOperation extends UntypedActor{
    ActorRef DBActorRef = this.context().actorOf(Props.create(DBActor.class),"DataBaseActor");

    SupervisorStrategy strategy = new OneForOneStrategy(500, Duration.create(1,TimeUnit.SECONDS),
            new Function<Throwable, Directive>(){

        @Override
        public Directive apply (Throwable throwable) throws Exception {
            if (throwable instanceof CommunicationException) {
                System.out.println("Resume called");
```

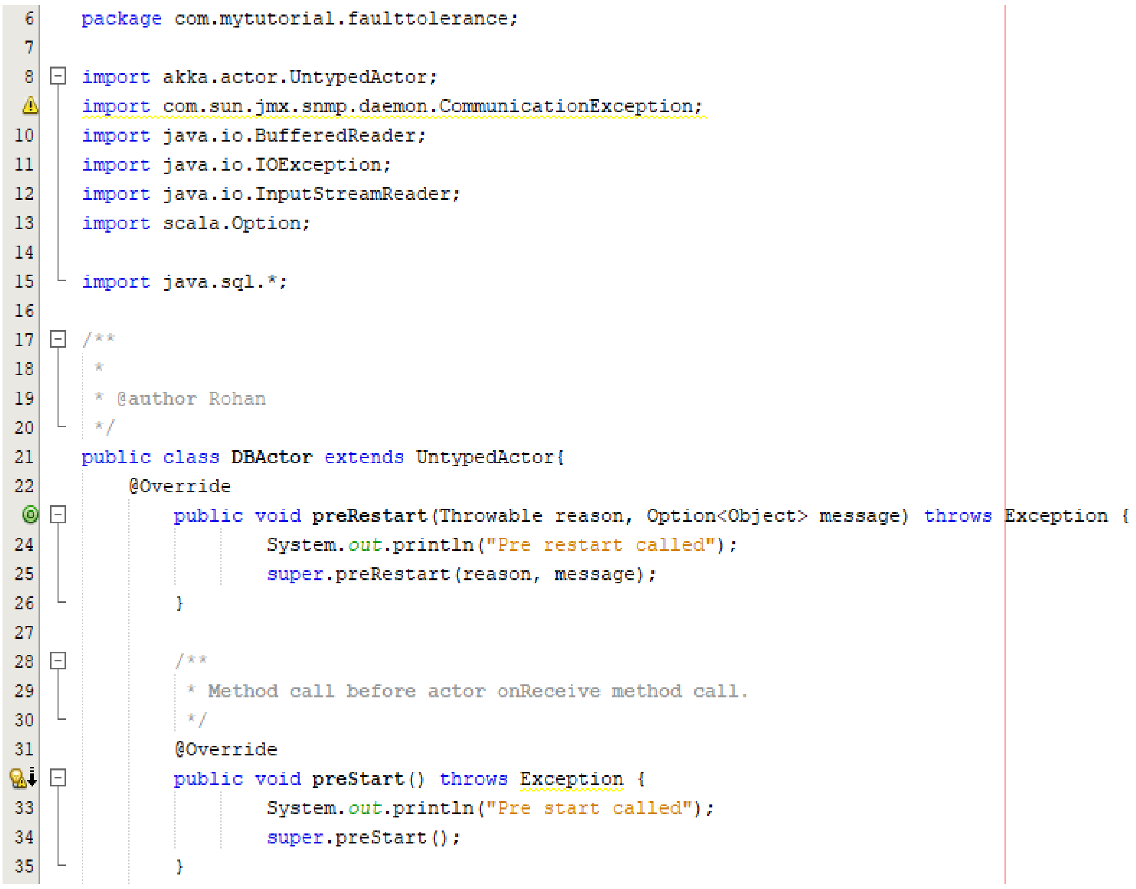


```
            if (throwable instanceof CommunicationException) {
                System.out.println("Resume called");
                return SupervisorStrategy.resume();
            }

            else {
                System.out.println("Stop called");
                return SupervisorStrategy.stop();
            }
        }
    });

    @Override
        public SupervisorStrategy supervisorStrategy() {
                return strategy;
        };


        @Override
        public void onReceive(Object arg0) throws Exception {

                DBActorRef.tell(new DBCredentials("jdbc:mysql://localhost/project", "root", ""), getSelf());
        }
}


class DBCredentials {
    String URL, User, Password;
    public DBCredentials(String url, String user, String password) {
```

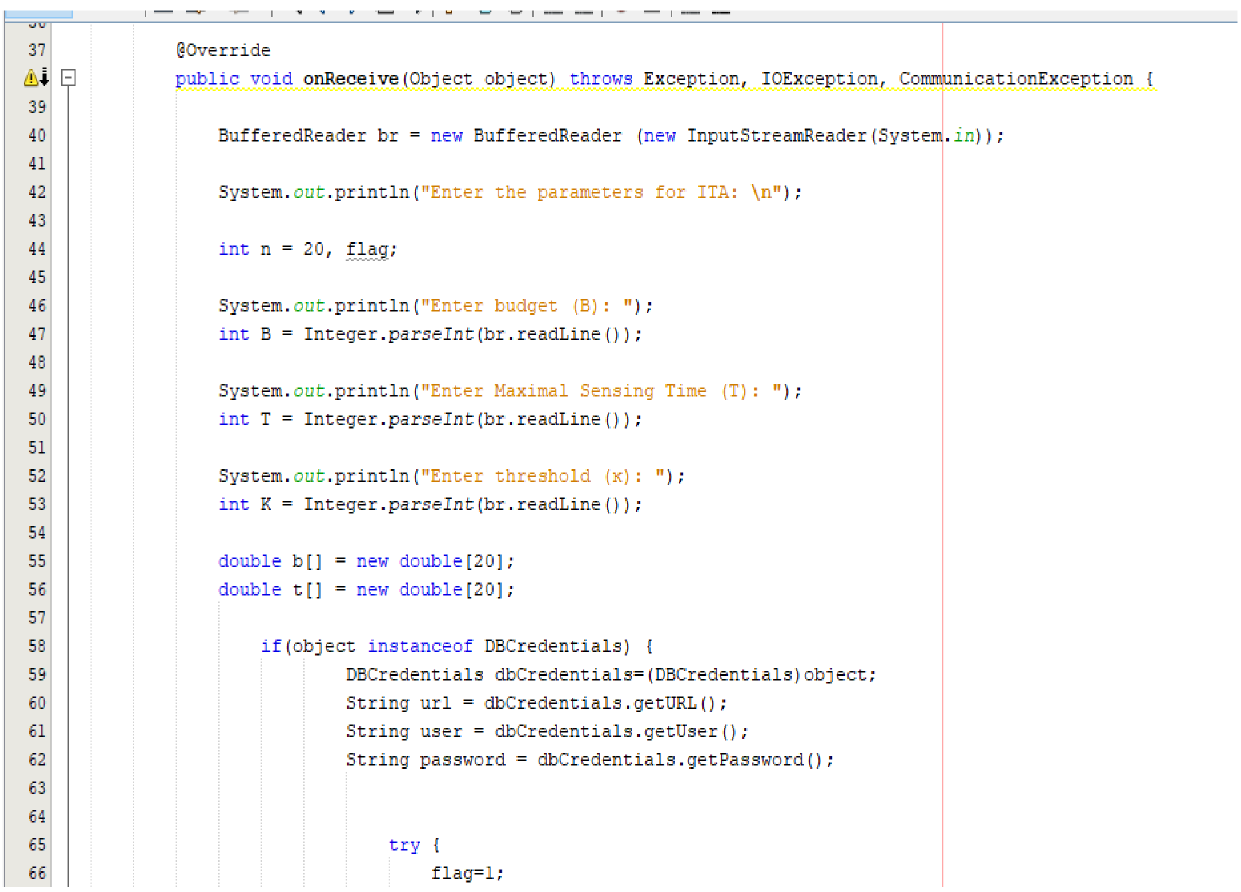


```


        @Override
        public void onReceive(Object arg0) throws Exception {

                DBActorRef.tell(new DBCredentials("jdbc:mysql://localhost/project", "root", ""), getSelf());
        }
}


class DBCredentials {
    String URL, User, Password;
    public DBCredentials(String url, String user, String password) {
        URL = url;
        User = user;
        Password = password;
    }

    public String getURL() {
        return URL;
    }
    public String getUser() {
        return User;
    }
    public String getPassword() {
        return Password;
    }
}
```

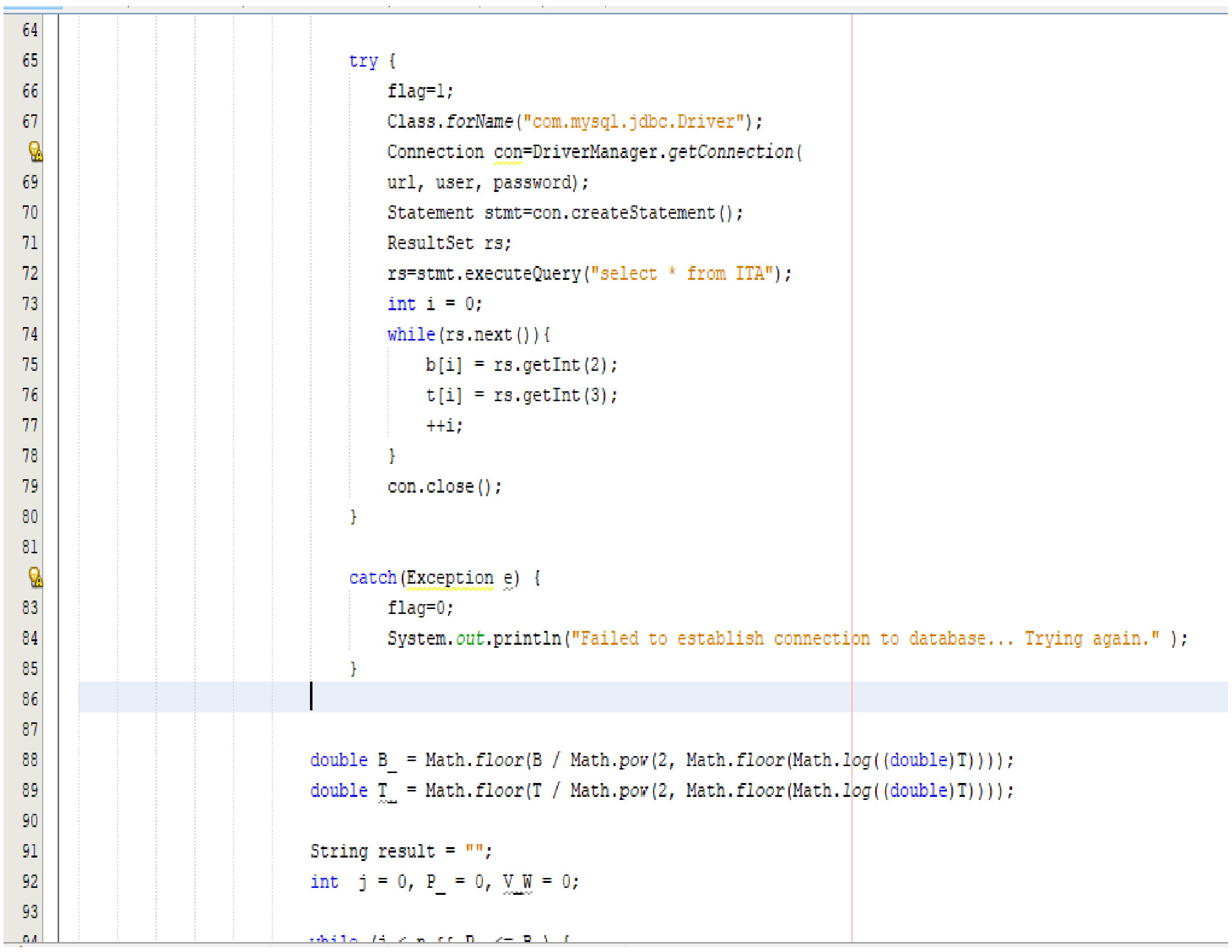
3. DBActor.java



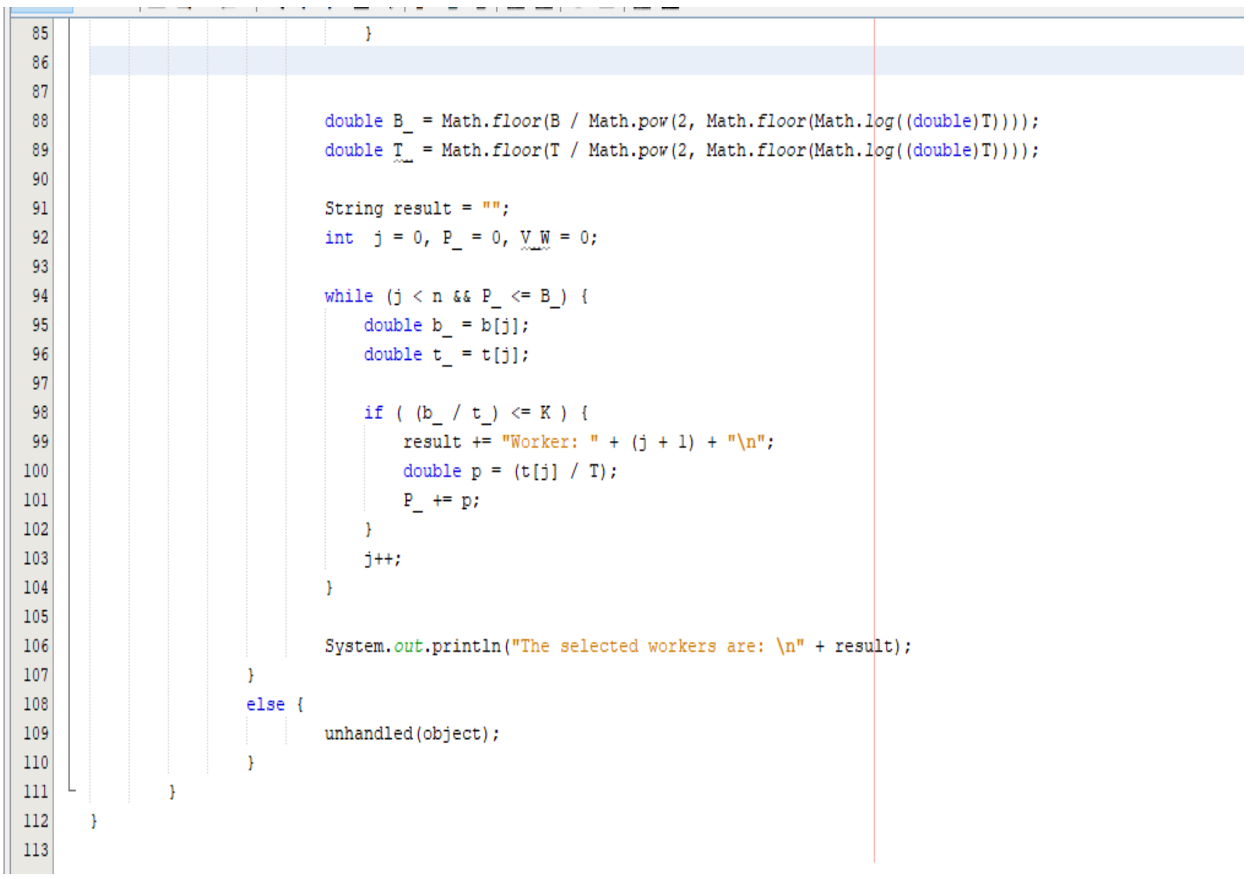
```
package com.mytutorial.faulttolerance;

import akka.actor.UntypedActor;
import com.sun.jmx.snmp.daemon.CommunicationException;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import scala.Option;

import java.sql.*;

/**
 *
 * @author Rohan
 */
public class DBActor extends UntypedActor{
    @Override
        public void preRestart(Throwable reason, Option<Object> message) throws Exception {
                System.out.println("Pre restart called");
                super.preRestart(reason, message);
        }

        /**
         * Method call before actor onReceive method call.
         */
        @Override
        public void preStart() throws Exception {
                System.out.println("Pre start called");
                super.preStart();
        }
```



```
        @Override
        public void onReceive(Object object) throws Exception, IOException, CommunicationException {

            BufferedReader br = new BufferedReader (new InputStreamReader(System.in));

            System.out.println("Enter the parameters for ITA: \n");

            int n = 20, flag;

            System.out.println("Enter budget (B): ");
            int B = Integer.parseInt(br.readLine());

            System.out.println("Enter Maximal Sensing Time (T): ");
            int T = Integer.parseInt(br.readLine());

            System.out.println("Enter threshold (K): ");
            int K = Integer.parseInt(br.readLine());

            double b[] = new double[20];
            double t[] = new double[20];

                if(object instanceof DBCredentials) {
                        DBCredentials dbCredentials=(DBCredentials)object;
                        String url = dbCredentials.getURL();
                        String user = dbCredentials.getUser();
                        String password = dbCredentials.getPassword();


                            try {
                                flag=1;
```



```

                            try {
                                flag=1;
                                Class.forName("com.mysql.jdbc.Driver");
                                Connection con=DriverManager.getConnection(
                                url, user, password);
                                Statement stmt=con.createStatement();
                                ResultSet rs;
                                rs=stmt.executeQuery("select * from ITA");
                                int i = 0;
                                while(rs.next()){
                                    b[i] = rs.getInt(2);
                                    t[i] = rs.getInt(3);
                                    ++i;
                                }
                                con.close();
                            }

                            catch(Exception e) {
                                flag=0;
                                System.out.println("Failed to establish connection to database... Trying again." );
                            }


                        double B_ = Math.floor(B / Math.pow(2, Math.floor(Math.log((double)T))));
                        double T_ = Math.floor(T / Math.pow(2, Math.floor(Math.log((double)T))));

                        String result = "";
                        int  j = 0, P_ = 0, V_W = 0;

```



```
                            }


                        double B_ = Math.floor(B / Math.pow(2, Math.floor(Math.log((double)T))));
                        double T_ = Math.floor(T / Math.pow(2, Math.floor(Math.log((double)T))));

                        String result = "";
                        int  j = 0, P_ = 0, V_W = 0;

                        while (j < n && P_ <= B_) {
                            double b_ = b[j];
                            double t_ = t[j];

                            if ( (b_ / t_) <= K ) {
                                result += "Worker: " + (j + 1) + "\n";
                                double p = (t[j] / T);
                                P_ += p;
                            }
                            j++;
                        }

                        System.out.println("The selected workers are: \n" + result);
                    }
                    else {
                        unhandled(object);
                    }
        }
}

```

34