ABSTRACT

Modern day usage of distributed systems has increased significantly. Apart from the list of advantages provided by distributed networks, there are few limitations across these networks. One of the key factors among these limitations are faults, which occur frequently across the networks. Among the possible faults, link failure and switching the mode of distributed networking are important. However, there are many fault recovery techniques. Most recovery techniques are static in nature, though, and they don’t have the ability to detect the dynamic link failures across the network.

The main aim of this project is to evaluate the role of a mobile agent-based fault tolerance system for the distributed network. In the proposed application, two servers and a master server are created. Each server can handle two clients, and faults like link failures and distributed network are inserted in this project. From the results, it is clear that the mobile agent can handle the dynamic link failures and improve the performance of the distributed network. Once the link failures and crashes are detected, a separate QoS module is invoked where metrics like Mean Time to Failure (MTTF) and Mean Time to Recover (MTTR) are calculated.
# Contents

1. INTRODUCTION ............................................................................................................. 7
   1.1 Distributed systems .................................................................................................. 9
   1.2 Different fault tolerant methods ............................................................................. 11
   1.3 Loop holes in existing system: ............................................................................ 12

2. HISTORY .......................................................................................................................... 13
   2.1 Distributed systems: .............................................................................................. 13
   2.2 Fault tolerance systems: ....................................................................................... 14
   2.3 Classification of faults: ........................................................................................ 16
      2.3.1 Timing faults: .................................................................................................. 17
      2.3.2 Omission failures: ........................................................................................ 17
      2.3.3 Crash faults: .................................................................................................. 17
   2.4 Failures across the distributed systems: ............................................................... 17
      2.4.1 Crash failures: ................................................................................................ 17
      2.4.2 Timing failures: .............................................................................................. 18
      2.4.3 Omission failures: ........................................................................................ 18
      2.4.4 Byzantine failures: ......................................................................................... 18
   2.5 Aims and Objectives ............................................................................................... 19
      2.5.1 Aim ............................................................................................................... 20
      2.5.2 Objectives ..................................................................................................... 20
3. Proposed System Design

3.1 Front end design

3.1.1 Master server

3.1.2 Server modules

3.1.3 Link status

3.2 Distributed network module

3.3 Client module

3.3.1 Send data

3.3.2 Receive data

3.4 Fault detection module

3.4.1 Fail the link

3.6 Mobile agent based module

3.7 QoS Metrics

3.8 Class Diagram:

Figure 2: Class Diagram

3.9 Use Case Diagram

3.10 Sequence Diagram

4. Functionalities of the Application

4.1 Module Description

4.1.1 Add client
4.1.2 Administration .................................................................................................................. 32
4.1.3 CommonUtil ..................................................................................................................... 33
4.1.5 LinkStatusForm: .............................................................................................................. 35
Figure 7: Link status screen .................................................................................................... Error! Bookmark not defined.
4.1.6 MasterServer .................................................................................................................... 36
4.1.7 MobileAgent ..................................................................................................................... 37
4.1.8 Receive data ..................................................................................................................... 38
4.1.9 Send Data ......................................................................................................................... 39
4.1.10 Quality of service (QoS): .............................................................................................. 41

5. TESTING AND IMPLEMENTATION .................................................................................... 44

5.1 Testing ................................................................................................................................. 44
5.1.1 White Box Testing: ......................................................................................................... 44
5.1.2 Black Box Testing: ......................................................................................................... 45
5.1.3 Unit Testing: .................................................................................................................... 45
5.1.4 Integration Testing: ........................................................................................................ 45
5.1.5 Validation Testing: .......................................................................................................... 46
5.1.6 Output Testing: ............................................................................................................... 46
5.1.7 Acceptance Testing: ....................................................................................................... 46
5.2 Testing Cases: ..................................................................................................................... 46

6. Conclusion ............................................................................................................................. 64
7. Future Work ......................................................................................................................... 66

BIBLIOGRAPHY AND REFERENCES ...................................................................................... 67
List of Figures

Figure 1: Proposed Architecture ................................................................. 21
Figure 2: Class Diagram ............................................................................... 27
Figure 3: Use case Diagram ...................................................................... 28
Figure 4: Sequence Diagram .................................................................... 30
Figure 5: Add client Screen ..................................................................... 32
Figure 6: Administration screen ............................................................... 33
Figure 7: Link Status Screen ................................................................... 36
Figure 8: Master server screen ................................................................. 37
Figure 9: Mobile agent screen ................................................................. 38
Figure 10: Receive Data screen ................................................................. 39
Figure 11: Send data screen ...................................................................... 40
Figure 12: Mean time to failure ................................................................. 42
Figure 13: Mean time to Repair ................................................................. 43
Figure 14: Administrators’ home page ...................................................... 47
Figure 15: Entering IP address ................................................................. 48
Figure 16: Entering a successful IP address .............................................. 48
Figure 17: Invalid IP address .................................................................... 49
Figure 20: An error – client name already entered ..................................... 51
Figure 21: Error – Enter a valid IP address ................................................. 52
Figure 22: Server’s home page ................................................................. 53
Figure 23: Sending data home page ......................................................... 54
Figure 24: Receiver’s home page .............................................................. 55
Figure 25: Add client home page ........................................................................................................ 56
Figure 26: Link status form .................................................................................................................. 57
Figure 27: Link status form to start or stop the client ........................................................................ 58
Figure 28: Mobile agent ....................................................................................................................... 59
Figure 29: QoS Home page ................................................................................................................. 60
Figure 30: Mean time to failure screen ............................................................................................... 61
Figure 31: Mean time to repair screen ................................................................................................. 62
1. INTRODUCTION

1.1 Distributed systems

Modern day usage of distributed systems has increased significantly. The main reason for this is the complex array of services and policies provided by the distributed systems. In general, different autonomous systems can be connected together to form a distributed architecture. These distributed systems provide access to the system resources like local memory. These systems also possess the ability to communicate with each other using message passing techniques [1].

A distributed system in general has a common goal, for example to solve a computational and logical problem, in which all the systems work collaboratively. Each and every autonomous system has an individual user working on a separate entity of the problem. Sharing of the resources is the key factor across any kind of distributed system. There are many techniques which help to share the resources among these autonomous systems.

Individual computational nodes or computers are monitored across the distributed systems to achieve the desired solution. Monitoring of these nodes includes recording all the possible cases of faults which may arise across the individual autonomous systems [2].

Distributed system should be designed in such a manner that they are able to tolerate the faults which may arise within the individual systems that form the group of autonomous systems. Wide ranges of applications are supported by these distributed systems. Some of the applications include: wireless sensor networks, telecommunication networks, Internet systems, peer to peer networks, routing algorithms, distributed database management systems, and network file systems [3]. Among all these applications, there are many autonomous entities working together
to achieve a common objective. The performance of each and every system should be considered in order to estimate the overall performance of the distributed system.

As there is an observed advancement in technology, new versions of distributed systems have arrived. These are known as dynamic distributed systems. More advanced features like dynamic data sharing and resource allocation are present in autonomous systems dynamically. There are many existing problems with dynamic distributed systems. The most common problems are with updating the systems with the changes in topology, handling the mobility of the processors, building tolerance of network failures, and handling the dynamic database requests [4].

Message passing is the most predominant task that runs across dynamic distributed systems. A lot of shared memory objects are used across this type of communication process. At the same time there is a lot of scope for faults across this process. Maintenance of the replicated database base objects is also a common issue which arises across dynamic distributed systems. The replication process may be interrupted due to the dynamic updates from any of the individual autonomous systems. This type of communication leads to faults in operation of the distributed system [5].

As discussed above, there are many problems and faults which occur across dynamic distributed systems. A perfect monitoring system is required always to handle these types of faults. There are many fault tolerant methods which help us to monitor dynamic distributed systems.
1.2 Different fault tolerant methods

Most distributed systems handle faults using agents. These agents have the ability to collect statistics and updates from the distributed systems. They send the inputs to the fault tolerant methods and the actual plan of action is decided based on the inputs from these agents. The working principle of these agents depends on the application range included across distributed systems. If the distributed system is all about mobile networks, these agents are known as mobile agents [6].

Mobile agents are responsible for monitoring the performance of the networks during varying conditions. They are responsible for updating different fault tolerance methods and making sure that the system is on track as per the predefined conditions and expected outcomes. In general, mobile agents are asynchronous and dynamic in nature. They have the ability to handle different monitoring requests from different aspects such as network failures across the large scale dynamic distributed systems. Many other faults are possible such as network topology changes or addition and deletion of the resources across the network.

In general there could be any number of domain managers spread across dynamic distributed systems. Handling such faults across domain managers is really a tedious job [7]. There are different mobile agent based fault tolerance methods to handle these failures. Most of them have proven to fail due to the heavy computational overhead in retrieving updates from the mobile agents. An efficient monitoring system is always required to handle failures. There are a number of issues which can make the monitoring system perform poorly due to continuous increase in the infrastructure. There could be some heterogeneous system updates and most of the managed resources become dynamic; thus the traditional static monitoring and mobile agents
are not sufficient. Dynamic mobile agents are always required to get the monitoring updates to ensure top performance of the systems [8].

In general a mobile agent is an autonomous and independent software module that can handle the individual user requests and gather the required monitoring information by visiting all the target nodes across the network. The most common advantages of mobile agents include dynamic network updates, handling the network traffic, delay and enabling the asynchronous execution of the programs.

1.3 Loop holes in existing system:

In general, mobile agents can be categorized into two different types: centralized and hierarchical. Centralized types are further divided into two categories: segment based and single based mechanism. In segment based mechanisms, a network is divided into two different child domains. In modern use, single monitoring agents are deployed to monitor the network. In a single based network, the whole network consists of a single agent to monitor the network. Due to this, the single agent mechanism is susceptible to scalability problems.

A hierarchical method visits the nodes across the network based on a predefined hierarchy [9]. The hierarchy method has the ability to solve the scalability issue seen in the single agent method. The hierarchical monitoring system will partition the network into different hierarchies and a new monitoring agent is deployed across each hierarchy. Thus the problem of scalability can be easily solved.
2. HISTORY

2.1 Distributed systems:

A distributed system can be generally defined as a set of computers which are independently connected together through a single network. These systems have the ability to communicate with other systems in a network. Distributed systems tend to perform a similar task in common. The knowledge about distributed systems is obtained through study known as “distributed computing”. Generally, the process of distributed computing is used to solve complex computational issues. These computational issues can be addressed by individual computers which are connected through a distributed computing network.

In distributed systems, a huge computation problem is initially divided into several small issues. Now, solving the issue is achieved by the autonomous computers which are individually connected across the system. Distributed systems have several features; uniqueness of systems can be noticed as every system has a local memory of its own. The most common feature in distributed systems is that a message passing technique is used by these systems to enable communication between the entities.

In distributed systems, all the autonomous systems are used for a common task and in addition to this, the systems will also have independent user personalities. Personal users can use these systems for their own purposes. In order to achieve a common goal, the computer connected in the distributed system will share some of the resources such as the information and memory while it performs the independent functions requested by the personal users.
Significant features of these systems are as follows:

- Fault tolerance is a default feature in distributed systems; because of this feature the faults and failures which exist in the network can be discovered.

- Distributed systems do not have a fixed structure and topology and they are not even known prior to joining the system. So, structures and topology can be changed according to the present requirements.

Across distributed systems, the autonomous computers will have restricted access to the other autonomous computers. Distribution of the task has to be done by all the systems in common, which is very similar to the techniques used in parallel computing. Even though these techniques are similar, there is an important difference. Parallel computing is a strongly coupled version of a distributed system. Communication between the computers in parallel computing is done via a particular approach as compared to the dynamic process used in distributed computing. The concept of shared memory is used in parallel computing in which memory is shared by all the computers in common; whereas distributed computing operates on the concept of local memory where all the systems have their own dedicated local memory. In parallel computing systems, a single memory and processor are used in common which means the system shares the available resources. In the case of distributed systems, there are several nodes available, each with their own memory and processor.

**2.2 Fault tolerance systems:**

A system can work perfectly with the help of some implementations generally known as a fault tolerant system. The performance of a fault tolerant system is estimated using the operation speed of the system. If the operation speed is decreased in the fault tolerant system, it indicates complex failure in the system. Generally, there are several factors where performance of a fault
tolerant system can be estimated. The most important factor among them is having the capacity to discover a fault in the system, the ability to address the fault, and finally make sure that the system works again. To be specific, fault tolerance of a system cannot be considered as a property because there are several other factors on which the system functionality is dependent.

The main entity that causes failures in systems is the functionality which is defined. Roll back and Roll forward are two approaches used by fault tolerant systems to respond against system errors. If an error is discovered, then the system will use the Roll forward approach to change the system to an appropriate state. The Roll back approach is entirely opposite to the roll forward approach. Using this approach, the system changes to an earlier version and then the system is forwarded to the current state using an error correction mechanism. There are several requirements that are necessary in developing a fault tolerant system. A few of those requirements are as follows:

- The entire system must not have a single failure in the network.
- Each and every failing aspect that is required must be isolated to the system failures that are possible
- According to the definition of the system, a high level fault contaminant is required.
- In order to use the Roll forward and Roll back approaches, different versions of the systems must be presented.
- In order to detect and prevent the existing failures in the system, it must be implemented with apt methods of failure recovery.

Therefore, all the above requirements show that design of fault tolerant systems must be done in such a way that they can handle all of the existing faults and errors in the system. The nature
of these fault tolerant systems is complex, based on the problems that are detected across the system.

Consider a scenario: A fault tolerant system is designed and implemented in a system. In a given case, it fails to identify a fault in the system. The system indicates that the level of complexity in the system is less than the risk that actually occurs in the system. In addition to the hardware level, equal importance is given to the application level because the performance of a fault tolerant system is based on both the levels, but the significant implementation takes place at the application level. The principles of replication and redundancy are generally used in fault tolerant systems to maintain higher versions and duplicate versions of the systems.

2.3 Classification of faults:

There are many faults observed in the systems. These are categorized based on severity and complexity levels. Faults are categorized into two types: transient and permanent. The transient ones are easily corrected by following simple steps and there is no need for special treatment. Permanent failures require a long time to recover from as they show long term effects on the systems. One advantage with these faults is that they can be easily traced out and solved by using the tolerant systems available.

Another way of categorizing faults is by the actual cause of failures. In this process, there are different types of failures such as design failures. Design failures occur due to improper designing of the system during the designing phase. Another important failure is operational failure and it occurs due to major issues such as processor crash and disk crashes. Irrespective of classification, given below are other faults and the fault tolerant systems to solve them.
2.3.1 Timing faults:

These types of faults are often seen when the service times out as mentioned earlier.

2.3.2 Omission failures:

These types of failures are observed when a particular service gets stuck in advance in a particular system.

2.3.3 Crash faults:

Crash failures are due to hardware crashes in a particular system especially as a result of processor crashes and disc failures.

Depending on these categories, estimation of their influence on the system is done. Fault tolerant systems are designed and implemented to trace out the faults. In the literature review, though, we discussed various fault tolerant systems. These play a vital role in distributed systems. Given below are the some of the faults that are found in the distributed systems, each of which are discussed in brief.

2.4 Failures across the distributed systems:

Distributed systems are capable of handling different types of failures. Few failures listed below are crucial and frequently occur in distributed systems.

2.4.1 Crash failures:

These failures occur on servers of the distributed systems. Once these failures are detected, the system operations are automatically suspended for a particular period. The best example is the operating system. Based on the operating system faults, fault tolerant system are designed and implemented.
2.4.2 Timing failures:

Timing failures also show their effect on the server of the distributed systems. In general, these faults occur when there is a gap between the client request time and the server response time because the server responds after a long time. Due to these timing failures, the clients may quit as they can’t get the response and it leads to sever operations.

2.4.3 Omission failures:

These failures also show their effect on servers. They are caused due to irregular responses from servers in the distribution system. Omission failures occur for various reasons but the vital ones are buffer overflow or sometimes the lack of server response. Due to these, failures rise in distributed system.

2.4.4 Byzantine failures:

Byzantine failures are often called arbitrary failures and are also seen in the server of distributed systems. Because of these failures, the server responds in an illogical way and creates trouble in distributed systems. This leads to unwanted outputs from the server. Sometimes, duplicate results are received and may lead to malicious entities. Due to these, the clients behave absurd which results to lots of distress.

In distributed systems, the recovery programs are designed based on the requirement gatherings and fault behaviors in the systems. A lot of cases against these failures are natural, and affects the exact solutions.

In some cases when the server goes down, the client will be unable to track the server in the distributed network. These situations can be solved by simple exception handling. But this is not recommended every time because of the complex coding logic of exception handling system.
In other cases, the client may not have access to the server in the distributed systems. This can be handled by writing simple time out logic. In this situation, the client needs to wait until the server responds with a positive signal. In some cases, more time is required. Still, it may be out of limit. But we have a chance of initiating the request. This is not preferred as it is not the best method since continuous time outs and reinitiating may lead to performance degradation. It provides scope for other side effects like idempotent operations (property that can be applied multiple times without changing the result beyond the initial application) in the distributed systems.

In many cases, servers are suddenly crashed. Clients send their request and expect the server to respond quickly. The best solution in this situation is to reconstruct the server and also the client request and help with the request. Sometimes, servers respond quickly and deliver the result but the results are lost before reaching a particular client. The server is usually unaware of it and continuously requests the client for response acknowledgement.

This kind of failure is solved by defining a time limit. If the maximum time is reached without a response, the server is alerted with a busy signal. There are different failures and their respective solutions are studied and applied with all possible research in distributed systems. Another important thing to remember is that these solutions are not up to the mark because they are confined to a certain range. In the case of major ones, these solutions may fail in the distributed systems.

2.5 Aims and Objectives
Following are the aims and objectives of this research
2.5.1 Aim

To develop a hierarchical based dynamic mobile agent to monitor the faults in dynamic distributed networks and also simulate the metrics in the distributed network.

2.5.2 Objectives

Following are the research objectives:-

1) To review and critically analyze different mobile agent based fault tolerant monitoring systems for dynamic distributed systems and evaluate the gaps in the literature.

2) To design a hierarchical network.

3) To develop a dynamic mobile agent based fault tolerant monitoring system that can collect the statistics from different nodes of the network.

4) To Develop a SQL database to maintain the routing tables and store the collected monitoring information.

5) To create a separate Quality of Service module that can calculate the Mean time to recover (MTTR) and Mean time to failure (MTTF).

6) To test the application of the project and evaluate the results.
3. Proposed System Design

A mobile agent based approach is developed to identify the failures across the dynamic distributed networks. A detailed explanation to the front end and database design is presented in figure 1.

![Figure 1: Proposed Architecture](image)

3.1 Front end design

A virtual dynamic distributed network is created using .Net. In order to create a distributed network, two server modules are created. A single server module can handle six clients in this design. A total of twelve clients are managed by two independent and distributed
servers. These two servers are controlled by a master server module. The actual implementation and design aspects to create these modules are explained as below

3.1.1 Master server

The master server is the key component in this process. It can maintain two independent and distributed servers. The master server holds the control of all the operations in the distributed network.

3.1.2 Server modules

Two servers are created and considered as the distributed servers in this module. Each server can handle six clients and in future more clients can be added as per the user requirements. This provision is also provided at the design level. A distributed server is represented by the server name like server A and “Start” and “Stop” radio buttons are provided. When the user clicks on the “Start” button, the server operations are initiated and the actual operations are listed below:

3.1.2.1 Send data

“Send data” is the primary operation available for both servers and this option is provided to the user. Whenever the user clicks on the “Send data” button, a browser window is opened and the user can select the desired file to be sent from the servers. Once the file is selected, the very next operation is to choose the client as each server can handle two or more servers. In general, the clients are identified with the client name or IP address. To simplify the concept, a unique number is assigned to the clients and this number is assumed to be an IP address of the client.

Once the client is selected, the send button is clicked, and then the sending process is initiated. The user can send the data to any of the clients by choosing the client's unique ID.
Once the data is sent to the client, it is stored in the database along with the number of packets sent and the client ID.

3.1.2.2 Receive data

Servers can receive the data from the clients while a separate option is provided to the users. Once this option is selected, the server can select the desired client from where the data should be received and once selected the corresponding data is stored across the server database table.

3.1.3 Link status

Link status from the server to the client and the client to the server can be checked using this module. Status of the link can be checked by selecting the link status option before sending or receiving data. If the status is “Connected”, then sending or receiving is done successfully. If the status is “Disconnected”, the corresponding process is terminated. Users can check the status of the link at any point in time. The link status is automatically set to “Stop” and this indicates that there are some failures across the distributed networks. An alternative fault tolerance system is developed to set the link status again to “Start” for a continuous communication mechanism.

3.2 Distributed network module

Making this application a distributed network, a separate distributed network module is created in this application and few assumptions are made in this context. In this module, some predefined metrics are defined such as the number of clients to be added and the amount of data to be sent. If these metrics are exceeded, then an alternative sever is selected to add the clients. The new server will process the client request. Following are the conditions to be applied in this context:-
1) A server can handle only six clients and if more clients are added to the server, these clients are automatically added to the next server.

2) If more clients need to be added to a server, an extra server is created and the new client is added to that server.

3) All the servers update the data to the master server.

3.3 Client module

Detailed explanations to the client module are given in this section. As mentioned above, a single server can handle not more than six clients and these clients can communicate with each other. Following are the key functions of the client module:-

3.3.1 Send data

Clients can send data to the desired server or client as per the requirements. Whenever the client wants to send the data, users are provided with the option to choose the desired destination. This destination can be either another client or another server. If, in any case, the distributed network conditions are violated, the user will get a warning message that the server is busy and now the users have to choose another server to process their request. While sending the data, users are provided with a browse button and a text area field so that they can choose the desired file or write some request at the sending option and they can click on the send button to send the data. Once the data is sent to the server or any other client, the corresponding details are stored across the database.

3.3.2 Receive data

Clients can receive data from any clients irrespective to the server. Whenever a client or server sends the data to a particular client, a separate link is provided to the users with respect to
check all the data received against that client. Whenever the user selects the link-status option, a table is generated showing all the details of the received data against the sender. The following link-status table contains columns like sender id, the date or time, the data sent, and the description of the data.

**3.4 Fault detection module**

As the main objective of this project is to evaluate a mobile agent based fault tolerant system for a distributed network, a fault detection module is introduced in this context. Now this module is initiated to detect the corresponding faults. Detailed explanations of the faults are as follows:-.

**3.4.1 Fail the link**

Links between the clients and the servers fail in this context. This can be achieved by the users by choosing the desired link between the desired server and the clients. A separate option is provided to the administrator. When the admin clicks on this link, administrators are provided with a list to choose the desired link to be failed. Once the admin selects the desired link, then the corresponding link is set to “Stop”. This is reflected in the link status module as well and it also updates the link status table as well. A separate failures table is also maintained and the corresponding details of these failures are maintained in the table. For example, the details like the actual nodes between the link and the time when the failure was occurred is maintained in this table.

**3.6 Mobile agent based module**

Once the failures are detected, the users can choose the mobile agent based fault tolerance option to detect the failures. A separate mobile agent is maintained in this application
that will detect the failures from the database table. Users can click on the run agent button to suggest the possible solutions to these failures and they are as explained below.

If there are any link failures in the network, this agent will suggest the alternative links. If there is a failure from server A to any other client, this mobile agent will scan the network and suggest alternative links to the client or server via other clients or servers. Now the users can choose the best alternative to set the link up, and once done the link status is set to on and the users can proceed with the communication. If there are any distributed network failures and users can’t use this option, the corresponding mobile agent can rectify this issue. In this case, the mobile agent will add more servers to the application. The new clients are added to those servers on a temporary basis and the agent will ensure the distributed network option works fine. The agent will delete these servers once the actual problem is rectified. The status of the distributed network is set to “Start” and the user can use this option in a similar manner.

3.7 QoS Metrics

Once the crash recovery is done, the overall quality of service is evaluated in this application. A few metrics like Mean time to failure (MTTF) and Mean time to recover (MTTR) are calculated. These metrics are calculated by comparing a normal crash failure detection system and the proposed mobile agent based approach and a clear evaluation is done.

3.8 Class Diagram:

A class diagram is generally known as static structure. It shows the functions of different modules in the system. They also show the operation performed by those modules. In figure 2, the upper part holds the name and the lower part holds the attributes.
Figure 2: Class Diagram
3.9 Use Case Diagram

The use case diagram represents the function of the project graphically with the help of the actors. In figure 3, we can clearly notice the dependencies.

![Use Case Diagram](image)

**Figure 3: Use case Diagram**

3.10 Sequence Diagram

Sequence diagram shows the interaction between different objects like server, mobile agent, and clients. It helps us to understand the sequence of operations performed in the system with one another. In figure 4, we can observe the messages exchanged between them.
Check Sts of all clients

Live means send the msg

Live means send the msg

Send Ack

Send Ack

GFDA means just receive the msg

GFDA means just receive the msg

Down means failure

Down means failure
4. Functionalities of the Application

The proposed design as mentioned above is implemented using .NET. All the required application interfaces are designed using the framework. The database is created in Oracle SQL. All the important classes and tables used along with the flow of the application are clearly described in this section.

4.1 Module Description

Every aspect across .NET programming is an object. These objects can be inherited in the form of class. These objects of the class can be created to initiate the core functionality used across the class. Following are the classes created in this project with respect to initiate the flow of the application. All the classes used across this application are created using the framework and the flow of the business logicas listed below:-

1) Add Client

2) Administration

3) CommonUtil

4) First

5) LinkStatusForm

6) Master server
7) Mobile Agent

8) Receive Data

9) Second Frame

10) SendDataForm

11) Server Module

4.1.1 Add client

Required logic to add the clients to the servers is given in this class. As per the design, few clients can be added to the server and the corresponding procedure is coded in this class. It works to create the logic. All the basic components like text fields, layouts and gaps are used to create the basic layout of the screen. A separate client ID is issued to the newly added client and assigned to the desired server. The newly added client identifies the server with the server id. Once the client is added, a new row is added to the client table and the corresponding query used here is “String countQuery="select * from client";”

Initially the number of clients is calculated. If this count is as per the defined handled, then only the client is added to the desired server. Otherwise, the client is added to the next available server. Once the required constraints are satisfied, the client is added to the client table and the rows are incremented by one against the table. Whenever the client is added to the server and the database is updated, the link status of the server and the client is also updated.

String query = "insert into client values (?,?,?,?) ";

String link_sts_query = "insert into link_status values(?,?,?) ";
In figure 5, when the client is added to the server, the corresponding client IP address and the client name are also updated against the database table.

![Master Server](image)

**Figure 5: Add client Screen**

### 4.1.2 Administration

Administration class is used to create the user interface for all the administration activities. The key administrative tasks used in this project are to start or stop server, check the status of the links, start the distribution mode of the network, and initiate the mobile agent for detecting the faults across the distributed networks.

The administrator can check the status of the links between the clients and the server at any point in time. A link status form is opened at the user interface and the administrators can
view the status of the desired link as seen in figure 6. A distributed network option is also available to the administrator and they can initiate the distributed mode of the network.

Admin is provided with a mobile agent button and the users can click on the button to start the mobile agent, where this agent collects the information related to the distributed network faults and suggests the options for the fault recovery options.

![Administration screen](image)

**Figure 6: Administration screen**

**4.1.3 CommonUtil**

This class is used to create the frequently used functions like the database functions. If at all a new instance of the database is required an object is created for this class and initiated to
dump the complete coding logic in this case. Following are the important methods used in this class.

**GetConnection**

This method is used to create the connection to the database and the basic JDBC drivers are used to create the connection to the database using the connection class across. Once the connection is established, the corresponding statements and other parameters are initiated. The Oracle thin driver is used to create the bridge to the database in this class as well as the user name and password to access the database.

**GetServerNameByServerId**

This method can be used to get the server name by providing the server as the required parameter. A separate global server is used as the required table and the server information is recorded in this table and thus the information is fetched from the database and the corresponding query used for this process is as given below

String serv_query="select * from globalserver where serv_id=''\"+serv_id+''";

**GetServerIdByServerName**

This method can be used to get the server name by providing the server name as the required parameter and the value is fetched from the global server database table. Following is the query used to get the required information from the table

String serv_query="select * from globalserver where serv_name=''\"+serv_name+''";
**GetClientIdByClient_name**

This method can be used to fetch the required client id with the help of client name and the client name is provided as the required parameter to the method. Below is the query is used to get the required client information with the help of client names

```
String serv_query="select * from client where client_name='"+client_name+"';
```

**GetClientNameByClientId**

This method can be used to get the client name by providing the client id as the required parameter and this name is fetched from the database table known as the client and the actual query is used in this context is as given below

```
String serv_query="select * from client where client_id='"+client_id+"';
```

**4.1.5 LinkStatusForm:**

As mentioned, the administrator can check the status of the link between the desired server and the client. The user interface is provided in such a way that the admin can select the desired link from the drop down list. All the ID’s of servers and clients are fetched from the
corresponding database. In figure 7, we can notice “Start and Stop” options are available where admin can select “Start or Stop” between the clients based on the situation.

![Link Status Screen](image)

**Figure 7: Link Status Screen**

4.1.6 **MasterServer**

The master server is initiated across this project. From the master server, admin can add the desired clients to respective servers. Admin issues unique IDs to the client. Once the user clicks on the desired button, the actual server operations are started. The actual action performed by the user is captured using the action events and the corresponding server is started.
Figure 8: Master server screen

In figure 8, it is clear that server A is selected to add the clients. The corresponding ID is tracked and the actual server module is started. This procedure is repeated for both the servers.

4.1.7 MobileAgent

This is the important module in this application. Mobile agent captures all the link failure information and displays it to the administrator. The administrator can set the status to “Start” again using this module. The user is also provided the required suggestion to check the alternative paths to communicate with the other clients. The link status information is fetched from the database. When the user authentication is failed, the connection is disconnected and a warning message is displayed to the admin.
From what is seen in figure 9, it is clear that the required information is gathered from the database. The status values are iterated in a loop using the result set object and the fetched values are displayed to the users in a tabular form. Required rows are added to this table using the information fetched from the database table. A vector object is used to store all the values retrieved from the database and the loop is iterated against this loop.

**4.1.8 Receive data**

As mentioned in the design of this project, clients and servers can receive the data from any other servers and clients. All the data received from any other client or server is displayed to the users in the form of a message. Database connections are established to gain
access to the information related to all the received data by a particular client or server. The data received is fetched from the table and corresponding values are displayed to the user.

![Data Sent](image)

**Figure 10: Receive Data screen**

In figure 10, we can observe how data is sent between the client “ankush” and “ssb”. The complete information is displayed to the admin like time, date, and message.

### 4.1.9 Send Data

Any client or server can send data to the desired client or server. A window is created across this section and the users can type the desired message to be sent. The client or server can
add the required message to the text area. The client can send any type of file like text, doc, pdf, impej. The data entered is sent to the desired client or server.

**Figure 11: Send data screen**

In figure 11, it is clear that data is sent to the selected client irrespective of the servers.

Following are the database tables used in this application

- Global server
- Client
- Link status
- Data packets.
4.1.10 Quality of service (QoS):

Quality of service (QoS) refers to several aspects of computer networks because it has the ability to provide best service to the selected traffic over different technologies. QoS helps to measure the traffic in the network. It can also be called a quality measure because it minimizes traffic delay, reduces input messages, and reduces packet drop. In this application, QoS is used to calculate mean time to failure and mean time to repair. This scenario helps us improve the efficiency and integrity between the server and clients.

Mean Time to Failure:

Mean time to failure is a method to predict the time the equipment will be available in the network. Mean time to failure helps us determine which systems were prone to failures. Mean time to failure is calculated by considering the total number of hours of service of all the systems divided by the total number of failures in the network. For example, devices are running for 9000 minutes and there are only 3 failures. This implies there can be at least one failure for every 3000 minutes.
In Figure 12, we can notice different values for the clients. The values in MTTF are presented in minutes. In Figure 12, we can notice clients “KKK” and “Siri” have the same values because both were present in the network at the same time.

**Mean time to Repair:**

Mean time to repair is a basic measure of the maintainability of repairable items. It helps to determine the average time required to repair a failed system. It can be calculated by considering the time from the occurrence of incident to the restoration of service. For example, let us consider client “KevinCheck” from Figure 13. In this illustration, we can notice the mean time to repair was seven minutes because the Client was disconnected at 10:00 am and restored the connection at 10:07 am.
Figure 13: Mean time to Repair

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Date</th>
<th>Client Name</th>
<th>MTTR</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>24-04-2012</td>
<td>kathik</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>24-04-2012</td>
<td>kkk</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>24-04-2012</td>
<td>siri</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>24-04-2012</td>
<td>ssa</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>24-04-2012</td>
<td>ssc</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>24-04-2012</td>
<td>ssd</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>24-04-2012</td>
<td>ankush</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>25-04-2012</td>
<td>kevinCheck</td>
<td>7</td>
</tr>
<tr>
<td>11</td>
<td>25-04-2012</td>
<td>ankush</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>25-04-2012</td>
<td>ssb</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>25-04-2012</td>
<td>home</td>
<td>1</td>
</tr>
</tbody>
</table>
5. TESTING AND IMPLEMENTATION

5.1 Testing

Testing plays a major role in developing the software efficiently without any errors. It plays an important role in software engineering. Testing is done to discover the errors in the existing system. It should be planned well in advance. It should be done at the end of every module. This advantage helps us to save time and money. However, there are different types of tests. Each test addresses a specific requirement.

Different types of tests:

1) White box testing.
2) Black box testing.
3) Unit testing.
4) Validation testing.
5) Integration testing.

5.1.1 White Box Testing:

Generally white box testing is also known as glass box testing. This method controls the internal structure of the procedural designs to evaluate the test cases.

1). It tests the internal parts of the structure whether they are working properly or not.

2). Considers different scenarios to analyze the logical issues.
5.1.2 Black Box Testing:

Generally black box testing is also known as behavioral testing. It performs the test without any basic knowledge of the internal structure of the program. It also has the ability to perform the test in requirements and specification phase by hiding it from the developer. It provides different inputs and responds to them, without any basic knowledge of the functioning of a program.

5.1.3 Unit Testing:

Generally unit testing is also known as module testing. In this phase it performs the test on small units to rectify the errors, at the initial stage itself.

   It performs the test at the development phase. This helps the developer make necessary changes as per the system requirements. This helps us save lots of time. Developers also have the ability to integrate the total program and perform the tests.

5.1.4 Integration Testing:

Integration performs testing on two different applications. These two different applications are integrated together using some techniques. It performs necessary actions and produces necessary error reports. As we know the project ends without errors after the unit testing. But in some situations there may be some errors in the interface. Integration tries to rectify those errors and correct them. With the help of the integration testing, we can evaluate project performance.
5.1.5 Validation Testing:

In the final stage of black box testing most of the errors are discovered and corrected. But still some tests should be performed. Those tests are generally known as Validation testing. In simple terms, Validation testing is completed when the project meets the total requirements of the customer.

5.1.6 Output Testing:

Output testing is performed to identify the correct output for the project. It will be a useless project if the output is wrong. This will help the developers, to identify the errors immediately before delivering to the customer.

5.1.7 Acceptance Testing:

This is the critical phase of the project. The development project should meet the customer's requirements fully. This phase also helps customers to identify whether the development project is user friendly or not.

5.2 Testing Cases:

The project begins with the administration module. Administrators are provided with a window. They have the ability to perform the required operations.
Administration:

Figure 14: Administrators’ home page

From the screen shown in figure 14, it can be observed that the administrator is provided with an option to start the Master Server from where the actual operations begin. The administrator is also provided with an option to check the status of the network using Link Status. At the same time, the administrator can start the mobile agent by clicking on the Mobile Agent option. The administrator can also calculate metrics. These metrics will be useful to increase the quality of the network.
Figure 15: Entering IP address

Figure 15 represents the servers. Each server has a unique IP address. The administrator has to enter the valid IP address to provide communication between the client and the server. The IP address usually varies in internet and intranet connections.

Figure 16: Entering a successful IP address

From figure 16, it can be noticed that the IP address had been successfully entered. A valid IP address is allocated to the respective servers to provide communication between servers and clients.
Figure 17: Invalid IP address

Figure 17 represents an invalid IP address. This error is due to the invalid IP address. Due to this error, the server cannot be initialized. This leads to communication issues between the server and the client. This issue can be fixed only by an administrator.

Mobile agent

Figure 18: Mobile agent home page
Figure 18 represents the mobile agent module. The mobile agent provides the required information related to the link failures. The corresponding information is displayed to the administrators in the form of a table. Corresponding suggestions with respect to these failures are also made. This module is actually invoked when there are faults across the network.

**Master server:**

![Master Server](image)

**Figure 19:** Master’s server home page
Figure 19 represents the Master server. In this module, when the administrator clicks on the Master Server, this particular window is displayed. From figure 19, we can notice that users are provided with the option to start the Server A and Server B. When the user selects server A, server A is started. When he clicks on server B, server B is also started. Two servers can be started at one point in time.

![Master Server Window](image)

**Figure 20: An error – client name already entered**

Figure 20 describes an error related to the client. From the above figure we can notice that the administrator is trying to enter the client details. But unfortunately those details already exist, so this leads to an error.
Figure 21: Error – Enter a valid IP address

Figure 21 describes an error related to the IP address. When the administrator enters an invalid IP address, this leads to an error. Figure 21's screen resembles the error. This error can be solved by entering a valid IP address.
Server Module:

Figure 22: Server’s home page

When the user selects server A, the screen seen in figure 22 is displayed to the user. From here the actual server operations are initiated. It can be noticed from figure 22 that users can select the desired client from the list of clients and proceed with the required operations. The users are provided with the options like send, receive.
Send Data:-

Figure 23: Sending data home page

Figure 23 represents the message sending screen between clients. From the figure 23, it is clear that the users can send the data from the desired server to the desired clients. Only active, connected networks are displayed to the user. The user needs to select the client from the drop down menu, and then data is sent to the selected client. The data sent is saved in the data sent table. Thus one can conclude that the data is sent to the desired client or server successfully.
Receiver Module:-

Figure 24: Receiver’s home page

All the data received is displayed in the form a table as shown in figure 24. This table contains the key columns like server name, client name, date, time and the actual message sent. This table can be used to track the log of receiving data by the administrator and also helps to check the level of communication between the clients and the corresponding servers.
Add client:

Figure 25: Add client home page

Figure 25 shows the procedure to add a client to a particular server. If the user selects server A to add the client, then server A is populated by default in the server field and the users can add the client IP address and the client name. Every client has a unique IP address. The client is added to the particular server if the server has not reached the limit to handle the required number of clients.
Figure 26: Link status form

Figure 26 represents the link status screen. Users can select the desired server and the corresponding client link to fail externally. This can be considered as the required fault across the distributed systems. If the server and the client are selected and the link fail option is clicked, then the corresponding link is failed. The communication among these nodes is no longer possible. Users should contact the administrator and select an alternative path for the communication. These paths can be restored with the help of the administrator.
Figure 27: Link status form to start or stop the client

The administrator has the ability to change the status of the link in the network. From figure 27, we can notice that “Failure” status cannot be changed to either start status or stop status. This reduces the input messages on the server.
Mobile Agent

Figure 28: Mobile agent

Figure 28 represents the mobile agent screen. It displays all the nodes that were connected and disconnected. It also displays the warning message if the user is unable to login successfully. This automatically leads to the link failure.

QoS Module

QoS is also known as Quality Measure. Quality of Service module helps the administrator to calculate the mean time to failure and mean time to repair. QoS helps to increase the quality of the network. This module helps the administrator to verify the data sent by the clients and the link status between the clients.
Figure 29: QoS Home page
Mean time to Failure (MTTF) module

**Figure 30: Mean time to failure screen**

Mean time to failure can be calculated by considering the total number of hours of service of all the systems divided by the total number of failures in the network crash. In figure 30 we can notice mean time to failure values of different clients. There may be variation between the values of some clients due to traffic in the network. For example, the clients are connected for 3000 minutes, and there are three failures in 3000 minutes. Now Mean time to failure is calculated by dividing the number of minutes the client accessed the network divided by the number of failures. For the above example, mean time to failure is 1000 minutes. This indicated there may be any failure occurring for every 1000 minutes of operation. The values are represented as minutes in mean time to failure.
MTTR module

**Figure 31: Mean time to repair screen**

Mean time to repair can be calculated by considering mean time before failure and mean time to failure. In the above figure 31 we can observe mean time to failure values of different clients. There may be variation between the values of some clients due to network or circuit failure. For example, the client is disconnected at 4 pm, and the connection is restored back at 4:30 pm. Now Mean time to repair is calculated by subtracting restored time from disconnected time. For the above example, mean time to repair is 30 minutes. The values are represented as minutes in mean time to repair.
Advantages

The proposed module helps us to access the data firstly. There is no need of the monitoring agent for each and every client. Each and every user can check the status of the clients or servers before sending the data. This helps us to save time, reduce the monitoring messages, and it also helps us to reduce the traffic on the network.

Summary

From the above results, it is clear that there could be many chances for faults to occur across the distributed networks. These faults can be in the range of link failures or distributed network failures. All these link failures can be identified by the mobile agent and displayed to the administrator at any point of time. This information is captured using a perfect database.

Now the administrators can handle these issues immediately by either setting the link status to ON or suggesting the users choose an alternative path for communication to happen. A few distributed network options such as adding the clients and servers and sending the information to the desired clients and servers are also provided for in this application.

From this study, the overall analysis is clear that more failures appear in the distributed network. A perfect mobile agent can handle these faults and set back the network of state. These mobile agents ensure each and every client is always connected to the desired server even in the case of faults. As the mobile agent can handle all the faults in an efficient manner, even all the loads on the servers are also reduced due to these mobile agents. The traditional handling of faults across the distributed network are resolved using the mobile agent based approach.
6. Conclusion

Distributed networks are widely used across the networking world these days. The main reason for this is due to their wide range of support to the distributed applications and nodes across the networks. Parallel computing and distributed computing are the previous modes of the distributed networks where distributed networks can handle multiple servers and clients.

A single server can handle multiple clients. Each and every client has the ability to process the data requests from any of the servers. The load on the master server is reduced a lot thanks to the help of distributed networks. There are a few limitations with the distributed networks as well and the key among them are the faults occurring across the network.

In general there are different faults that may occur across the distributed networks. The most important among them are link failures and when the distributed mode of the network is down for some time. In these scenarios, the communications among the nodes of the distributed networks halts. Even adding the clients and servers to the distributed networks is stopped. Thus, the overall performance of the distributed network affects a lot. There are many fault tolerant techniques available across the literature, but most of them are static in nature and can’t gather the required information with respect to the link failures.

The main aim of this project is to evaluate the role of mobile agents in detecting the faults across the distributed networks and incur the fault tolerance techniques across the networks. To prove the proposed concept, Java based application is used and the corresponding design is given in this project.

A master server is used in this project and this server can handle two servers. Each server considered can handle a maximum of six clients. A distributed mode of the network is also
provided in this project, where the new clients are added to the next server if the maximum limit of the servers is reached, and in this way the load on the server is reduced significantly.

From the detailed analysis of the results, it is clear that there could be many chances for the faults to be occurred across the distributed networks and these faults can be in the range of link failures or distributed network failures. All these link failures are identified by the mobile agent and displayed to the administrator at any point in time. This information is captured using a perfect database. Now the administrators can handle these issues immediately by either setting the link status to ON or suggesting the users choose an alternative path for the communication to happen.

A few distributed network options such as adding the clients and servers and sending the information to the desired clients and servers are also provided for in this application. From the overall analysis, it is clear that distributed networks are prone to faults, but a perfect mobile agent can handle these faults and set the network back into operational state, ensuring each and every client is always connected to the desired server. This is accomplished even in the case of faults as the mobile agent can handle all the faults in an efficient manner. The load on the servers is also reduced due to this mobile agent. The traditional handling of faults across the distributed network are resolved using the mobile agent based approach.
7. Future Work

Apart from the work completed to prove the importance of the mobile agent in detecting the faults across the distributed network, future work can be done in this application to improve the scope of the application.

- A detailed evaluation of the mobile agent methodology can be done in future to improve the quality of finding and detecting the faults across the distributed networks.
- Few real time simulation tools can be used to evaluate and compare different agent based fault tolerance techniques across the distributed networks.
- More number of servers and clients can be added to the application to understand the real impact of load on faults across the distributed networks.
BIBLIOGRAPHY AND REFERENCES


Last accessed 29th March 2012.


Available: http://otago.ourarchive.ac.nz/bitstream/handle/10523/1184/SENG480Diss_AnhThuLe.pdf

Last accessed 29th March 2012.


Last accessed 29th March 2012.


Last accessed 29th March 2012.


Last accessed 29th March 2012.


Last accessed 30th March 2012.


Last accessed 30th March 2012.


Last accessed 30th March 2012.


Last accessed 30th March 2012.