LOW LATENCY FAULT TOLERANCE SYSTEM FOR DISTRIBUTED APPLICATIONS

CHU SANDY KYAW

M.C.Sc. **JUNE 2022**

LOW LATENCY FAULT TOLERANCE SYSTEM FOR DISTRIBUTED APPLICATIONS

BY

CHU SANDY KYAW

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Computer Science

(M.C.Sc.)

University of Computer Studies, Yangon

JUNE 2022

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my sincere thanks to those who helped me with various aspects of conducting research and writing this thesis. To complete this thesis, many things are needed like my hard work as well as the support of many people.

First and foremost, I would like to express my deepest gratitude and my thanks to **Dr. Mie Mie Khin**, Rector of the University of Computer Studies, Yangon, for her kind permission to submit this thesis.

I would like to express my appreciation to **Dr. Si Si Mar Win and Dr. Tin Zar Thaw**, Professors of Faculty of Computer Science, the University of Computer Studies, Yangon, for their superior suggestion, administrative support, and encouragement during my academic study.

My thanks and regards go to my supervisor, **Daw Marlar Win Khin**, Associate Professor, the Department of Mathematics, the University of Computer Studies, Yangon, for her support, guidance, supervision, patience, and encouragement during the period of study towards completion of this thesis.

I also wish to express my deepest gratitude to **Daw Aye Aye Khine,** Associate Professor, the Department of English, the University of Computer Studies, Yangon, for her editing this thesis from the language point of view.

Moreover, I would like to extend my thanks to all my teachers who taught me throughout the master's degree course and my friends for their cooperation.

I especially thank my parents, all my colleagues, and friends for their encouragement and help during my thesis.

STATEMENT OF ORIGINALITY

I hereby certify that the work embodied in this thesis is the result of original research and has been submitted for a higher degree to any other University or Institution.

…… ……

Date Chu Sandy Kyaw

Abstract

The Low Latency Fault Tolerance (LLFT) system provides fault tolerance for distributed applications within a wide-area network, using a leader-follower replication strategy. LLFT provides application-transparent replication, with strong replica consistency, for applications that involve multiple interacting processes or threads. The LLFT Messaging Protocol provides reliable, totally ordered message delivery by employing a group multicast, where the message ordering is determined by the primary replica in the destination group. The Leader-Determined Membership Protocol provides reconfiguration and recovery when a replica becomes faulty and when a replica joins or leaves a group, where the membership of the group is determined by the primary replica. LLFT can operate in the common industrial case where there is a primary replica and one or more backup replicas. The LLFT system achieves low latency message delivery during normal operation and low latency reconfiguration and recovery when a fault occurs.

As in other fault tolerance systems, the replicas of a process form a process group. One replica in the group is the primary, and the other replicas are the backups. The primary multicasts messages to a destination group over a virtual connection. The primary in the destination group orders the messages, performs the operations, produces ordering information for non-deterministic operations, and supplies ordering information to its backups. Thus, the backups can perform the same operations in the same order and obtain the same results as the primary. If the primary fails, a new primary is chosen deterministically and the new primary determines the membership of the group. LLFT operates within the usual asynchronous model, but with timing-based fault detectors. The assumptions of eventual reliable communication and sufficient replicati on enable LLFT to maintain a single consistent infinite computation, despite crash, timing, and partitioning faults

The proposed LLFT system is emphasized on the Furniture Ordering Management System.

TABLE OF CONTENTS

Page

ACKNOWLEDGEMENTS i

STATEMENT OF ORIGINALITY ii

ABSTRACT iii

TABLE OF CONTENTS iv

LIST OF FIGURES vi

CHAPTER 1 INTRODUCTION

1.1 Objective of the Thesis 2

1.2 The Related Work 2

1.3 Organization of the Thesis 3

CHAPTER 2 BACKGROUND THEORY

2.1 Consistency Models 4

2.2 Consistency Protocols 6

2.3 Primary Replica Based Protocol 6

2.4 Replicated Write Protocol 6

2.4.1 Active Replication 6

2.4.2 Quorum Based 6

2.5 The Concept of Replication 7

2.6 Database Replication 8

2.7 Choosing Database Replication 9

2.8 The Ugly of Database Replication 9

2.9 Methods of Performing Database Replication 10

2.9.1 Snapshot Replication 11

2.9.2 Merging Replication 12
2.9.3 Transactional Replication 13
2.10 Transaction 15
2.11 Distributed Transactional Management 15
CHAPTER 3 ACTIVE AND PASSIVE REPLICATIONS USING MULTICAST
3.1 Replication 17
3.2 Functional Model 19
3.3 Active Replication 23
3.4 Passive Replication 25
CHAPTER 4 THE SYSTEM DESIGN AND IMPLEMENTATION
4.1 Example Operation of Semi- active and Semi-passive in Proposed System 30
4.2 The Proposed Low Latency Fault Tolerance Algorithm 33
4.3 Implementation of the System 35
4.3.1 Adding New Product and New Category at the Admin Site 36
4.3.2 Furniture Ordering Process at Customer Site 47
4.3.3 Confirmed Order Saving at Admin Site 40
CHAPTER 5 CONCLUSION AND FURTHER EXTENSIONS
5.1 Advantages of the System 42
5.2 Limitations and Further Extensions 42
AUTHOR'S PUBLICATION 44
REFERENCES 45

LIST OF FIGURES

Figure	Page
Figure 2.1 Snapshot Replication	12
Figure 2.2 Transactional Replication	15
Figure 2.3 Flat Distributed Transaction	17
Figure 2.4 Nested Distributed Transaction	17
Figure 3.1 Functional Model with the five phases	20
Figure 3.2 Active Replication	25
Figure 3.3 General Model of Replica Management	26
Figure 3.4 Passive Replication	26
Figure 3.5 General Model of Replica Management (In Passive)	27
Figure 4.1 The System Flow	30
Figure 4.2 The Detail Process Flow of LLFT in Proposed System	31
Figure 4.3 Dashboard –Primary Down (Admin Site)	32
Figure 4.4 Dashboard –Two Replicas – Ready (Admin Site)	33
Figure 4.5 Sequence Diagram of System	33
Figure 4.6 Admin Login	36
Figure 4.7 Customer Login Page	36
Figure 4.8 Add New Product (Admin Site)	37
Figure 4.9 Category List (Admin Site)	37
Figure 4.10 Home Page (Customer Site)	38
Figure 4.11 Category List (Admin Site)	38
Figure 4.12 Product List to Order (Customer Site)	39
Figure 4.13 Ordering Page (Customer Site)	40
Figure 4.14 Confirmed Order List (Customer Site)	40
Figure 4.15 Customer List (Admin Site)	41
Figure 4.16 Order List (Admin Site)	41
Figure 4.17 About Page of the System	42

CHAPTER 1
INTRODUCTION

Nowadays, business becomes increasingly large, and it influences to open new branch workplaces in various areas. In this manner, taking care of the request for each dispersed branch is significant. A conveyed distributed database system (DDBS) is an assortment of a few consistently related data sets which are genuinely circulated in various PCs (generally called locales) over a PC organization.

The Low Latency Fault Tolerance (LLFT) system gives adaptation to internal failure to circulated applications, utilizing an exceptionally upgraded pioneer devotee replication methodology, to accomplish significantly lower dormancy and more quick reactions than existing gathering correspondence frameworks. LLFT gives adaptation to internal failure to disseminated applications over a wide-region organization. One imitation in the gathering is the primary, and different reproductions are the reinforcements.

The primary multicasts messages to an objective gathering over an association. The primary in the objective gathering orders the messages, plays out the tasks, produces requesting data, and supplies that requesting data to its reinforcements. Subsequently, the reinforcements can play out similar tasks in a similar request and get similar outcomes as the essential. If the primary fizzles, a new primary is picked deterministically and the new primary decides the participation of the gathering [2].

In LLFT, the handling and correspondence are offbeat, yet the issue identifiers force timing limits. The suppositions of possible solid correspondence and sufficient replication empower LLFT to keep a solitary predictable infinite calculation, despite crash. LLFT utilizes the pioneer devotee technique to lay out a complete request of messages, and a steady gathering participation.

1.1 Objectives of the Thesis

The objectives of the thesis are as follows:

- To ensure consistency between redundant resources for improving reliability, fault-tolerance, or accessibility.
- To maintain a consistent computation, despite faults and asynchronous communication.
- To provide reliable, totally ordered message delivery in the destination group.

- To achieve low latency message delivery during normal operation.
- To maintain low latency reconfiguration and recovery when a fault occurs.
- To support the high response even in the case of the primary fail.

1.2 The Related Work

The connected works of replication are examined in this meeting. Andre Brito and Pascal Felber zeroed in on dynamic replication as a way to deal with give adaptation to non-critical failure to Event Stream Processing (ESP) administrators. All the more exactly, they tended to the exhibition expenses of dynamic replication for administrators in disseminated ESP applications. They utilized a hypothesis component in view of Software Transactional Memory (STM) to accomplish the accompanying objectives: (I) empower copies to gain ground utilizing hopeful conveyance; (ii) empower early sending of speculative calculation results; (iii) empower dynamic replication of multi-strung administrators utilizing value-based executions. Trial assessment demonstrated the way that, utilizing this blend of components, one can carry out profoundly effective issue lenient ESP administrators [3]. They have proposed new methods to effectively uphold dynamic replication in shortcoming lenient disseminated occasion handling frameworks. By utilizing a STM-based theory instrument, they permitted hubs to hopefully begin handling occasions before their last conveyance (before their separate request is known with conviction) [2].

Yair Amir and Ciprian Tutu introduced a total calculation for information base replication over partitionable organizations sophistically using bunch correspondence and demonstrated its rightness. Their evasion of the requirement for start to finish affirmation per activity added to prevalent execution. They told the best way to consolidate online launch of new reproductions and super durable evacuation of existing imitations. They likewise showed how to effectively uphold different kinds of utilizations that expected different semantics [11].

1.3. Organization of the Thesis

This thesis is organized in five chapters. Chapter 1 describes the introduction of concurrency control, objectives of the thesis, Related Works of the System and thesis organization. Chapter 2 presents the background theory of the replication system and the replication techniques of active, and passive are discussed in Chapter 3. Chapter 4 presents Design and Implementation

of the proposed system. Conclusion, Limitation and Further Extension of this thesis are presented in Chapter 5.

CHAPTER 2
BACKGROUND THEORY

An important issue in distributed systems is the replication of data. Data are generally replicated to enhance reliability or improve performance. One of the major problems is keeping replicas consistent. Informally, this means that when one copy is updated, we need to ensure that the other copies are updated as well; otherwise, the replicas will no longer be the same.

2.1 Consistency Models

The distinctions of consistency models are thought of. One of the significant properties of a framework configuration is consistency model. This property can normally be presented in relations of an express that can be valid or bogus for various executions. Consistency models are alluded to as the agreements among interaction and information for guaranteeing rightness of the situation. Consistency models are introduced through various consistency rules to be fulfilled by appraisals of tasks. For standard consistency states of the ACID properties, there exist a few techniques for consistency ensure. In ACID consistency strategy, data set is in a reliable state when an exchange is done. In the client level, there are four parts:

- DS (Database System) is a storage system.
- Dad (Discharge Abstract Database) is the cycle activity for each read or composed by DS.
- PB (Priority Blocking) is sovereign of cycle PA (Priority Abort) that plays out each perused and composed activity from the DS.
- PC (Priority Control) is sovereign of cycle PA that plays out each perused and composed activity from the DS.

In the client level consistency, it is vital that how and when an eyewitness is happened. The PA, PB and PC processes see refreshes with an information thing in the capacity framework. There are two consistency types like Data-Centric consistency and Client-Centric consistency [1].

In Data-Centric consistency models, there are:

- **Severe consistency**: All the A, B and C send back the aftereffect of update esteem when the update methodology is finished.

- **Consecutive consistency**: The degree of successive consistency is lower than severe consistency. Each read and compose activity is performed by all reproductions on their information thing x successively. Additionally, each discrete system activities execute the recognized request.
- **Causal consistency**: This consistency is more fragile than severe and successive consistency. On the off chance that exchange T1(Transaction1) is affected or caused on a prior exchange T2(Transaction2), every reproduction ought to be first see T2, and afterward see T1.
- **FIFO (First In First Out) consistency**: FIFO consistency is loose to execute in light of the fact that it is being ensured at least two composes from a solitary source should show up in the request gave. Fundamentally, this really intends that with FIFO consistency, all composes produced by various cycles are simultaneous [6].

In Client-Centric consistency models, there are:

- **Inevitable consistency**: This model ensures on the off chance that updates are finished to the thing at last, all gets to on this information thing send back the past refreshed esteem.
- **Monotonic Reads**: In this model if an activity peruses the information thing x, in every case each following read procedure on information thing x send back same worth x or a later worth.
- **Monotonic Writes**: In this model if an activity composes on the information thing x, in every case each following compose procedure on information thing x comes after related compose procedure on the information thing x.
- **Peruse your-Write**: The consequence of a compose procedure on the information thing x generally will be acknowledged by a following read procedure on x by a similar worth.
- **Compose follow read**: In this model the impact of a composed procedure on an information thing x following a past read procedure on information thing x by the very esteem that is ensured to happen on something similar or a later worth of x that was perused [6].

2.2 Consistency Protocols

A consistency protocol explains as an implementation of a specific consistency model.

2.3 Primary Replica Based Protocol

In this convention, all composed tasks to an information thing x is gone to by one explicit reproduction that called essential copy. This essential copy is responsible for refreshing different reproductions; the client simply participates by this essential imitation. Two necessities ought to be occurred for this liberal of convention:

- All read and composed tasks for refreshing an information thing x ought to spread and be executed all reproductions sooner or later.
- These activities ought to be executed in a similar request [9].

2.4 Replicated Write Protocol

In this protocol, each write operations are sent to each replica to update procedure. There are two types for replicated write protocols.

2.4.1 Active Replication

In active replication, every imitation contains a corresponding technique that vehicles out the update tasks. Not at all like different conventions, update tasks are regularly engendered through the compose activity. This engendering causes the activity is shipped off every copy. Likewise, it is necessary a complete request for all compose tasks that every imitation executes a similar request of compose orders [6].

2.4.2 Quorum Based

This convention indicates that the clients get the approval of a few servers before any perusing or composing an imitated information thing x. For instance, the composed activities just need to be executed on piece of all copies before return to the client. It utilizes decisions to keep away from compose read struggle and compose struggle:

- R is the quantity of imitations of every information thing.
- Rr is number of imitations that a client ought to contact by them for perusing a worth.
- Rw is number of imitations that a client ought to contact by them for composing a worth.
- For forestalling the Write-Endlessly compose Read clashes,

- Rr + Rw > R and Rw + Rw > R ought to be fulfilled. [1]

2.5 The Concept of Replication

Replication addresses the most common way of dividing data to guarantee consistency among excess assets, like programming or equipment parts, to further develop unwavering quality, adaptation to non-critical failure, or availability. It very well may be information replication assuming similar information is put away on numerous capacity gadgets or calculation replication if a similar figuring task is executed commonly. The admittance to a duplicated substance is commonly uniform with admittance to a solitary, non-recreated element. The actual replication ought to be straightforward to an outer client. What's more, in a disappointment situation, a failover of reproductions is concealed however much as could reasonably be expected. In frameworks that reproduce information the actual replication is either dynamic or latent.

A functioning replication when a similar solicitation is handled at each imitated occurrence and about latent replication when each solicitation is handled on a solitary reproduction and afterward its state is moved to different copies. On the off chance that whenever one expert imitation is assigned to deal with every one of the solicitations, then the essential reinforcement plot (ace slave conspire) prevalent in high-accessibility groups. On the opposite side, in the event that any imitation cycles a solicitation and, conveys another state, then this is a multi-essential plan (called multi-ace in the data set field). [8] Even though, the course of Data Replication it's utilized to make occurrences of the equivalent or portions of similar information, it with the course of reinforcement since imitations are every now and again refreshed and immediately lose any verifiable state. Reinforcement then again saves a duplicate of information unaltered for an extensive stretch of time.

2.6 Database Replication

Database replication is the most common way of making and keeping up with various examples of similar data set and the method involved with sharing information or data set plan changes between data sets in various areas without duplicating the whole data set.

In many executions of data set replication, one data set server keeps up with the expert duplicate of the data set and the extra data set servers keep up with slave duplicates of the data set. The at least two duplicates of a solitary data set stay synchronized. [11] The first data set is known

as a Design Master and each duplicate of the data set is known as an imitation. Together, the Design Master and the reproductions make up an imitation set. There is just a single Design Master in a reproduction set. Synchronization is the method involved with guaranteeing that each duplicate of the data set contains similar items and information. At the point when client synchronizes the copies in an imitation set, just the information that has changed is refreshed.

Information based composes are shipped off the expert data set server and are then duplicated by the slave data set servers. Data set peruses are split between all the data set servers, which brings about a huge execution advantage because of burden sharing. The exchange handling burden can be appropriated among every one of the copies in the framework. This prompts a bigger throughput (since questions and peruse tasks don't change the data set state, they can be freely executed in one copy just) and a more limited reaction times for inquiries (since inquiries can be executed exclusively in one imitation, which is as a rule in a similar area as the client, and with no extra correspondence among the reproductions). [8] likewise, information-based replication can likewise further develop accessibility because the slave data set servers can be designed to assume control over the expert job in the event that the expert data set server becomes inaccessible. [3] If one replica crushes because of a product or equipment disappointment, the leftover copies can in any case cycle, while concentrated data set framework turns out to be totally inaccessible after only one accident [3].

2.7 Choosing Database Replication

Carrying out and keeping up with replication probably will not be a basic recommendation. If various data set servers that should be engaged with different sorts of replication, a straightforward errand can immediately become complicated.

Carrying out replication can likewise be convoluted by the application engineering. Notwithstanding, there are various situations in which replication can be used [3].

Data set replication is appropriate to business arrangements that need to:

- Divide information between distant workplaces
- Divide information between scattered clients
- Make server information more open
- Appropriate arrangement refreshes
- Back up information
- Give Internet or intranet replication [5].

2.8 The Ugly of Database Replication

Despite the fact that data set replication has many advantages and can tackle numerous issues in appropriated data set handling, the way that in certain circumstances replication is not so great. Information base Replication isn't suggested if:

There are incessant updates of existing records at numerous copies

- **Incessant updates of existing records at numerous copies**: Arrangements that have an enormous number of record refreshes in various imitations are probably going to have more record clashes than arrangements that basically embed new keeps in a data set. In the event that changes are made to similar record by various clients and simultaneously then record clashes will show up. This can be ongoing consuming on the grounds that the contentions should be settled physically.
- **Information consistency is always crucial**: Arrangements that depend on data being right consistently, for example, reserves move, carrier reservations, and the following of bundle shipments, ordinarily utilize an exchange technique. Despite the fact that exchanges can be handled inside a copy, there is no help for handling exchanges across reproductions. The data traded between reproductions during synchronization is the aftereffect of the exchange, not the actual exchange.
- **Above connected with an extra handling and correspondence**: The imitations require extra correspondence to guarantee that adjustments are applied to all the data set duplicates, which expands the heap on the machines and in the correspondence organization, in this way debases the general framework execution.[2]
- **Framework intricacy**: Synchronization of the data set duplicates among reproductions requires use of cutting-edge correspondence and exchange handling calculations. [2]

2.9 Methods of Performing Database Replication

Data set replication can be acted in not less than three unique ways:

- Depiction replication: Data on one information base server is clearly duplicated to another data set server, or to one more data set on a similar server.
- Consolidating replication: Data from at least two information bases is joined into a solitary data set.

- Conditional replication: Users acquire total starting duplicates of the data set and afterward get occasional updates as information changes [2].

2.9.1 Snapshot Replication

This sort of Database Replication is one of the most straightforward techniques to set up, and maybe the simplest to comprehend.

The preview replication strategy capacities by occasionally sending information in mass configuration. Normally it is involved while the buying in servers can work in read-just climate, and furthermore while the buying in server can work for quite a while without refreshed information. Working without refreshed information for a while is alluded to as idleness.

For instance, a retail location involves replication for the purpose of keeping a precise stock all through the locale. Since the stock can be overseen on a week by week or even month to month premise, the retail locations can work without refreshing the focal server for quite a long time at a time. This situation has a serious level of dormancy and is an ideal contender for depiction replication.

Extra motivations to utilize this sort of replication incorporate situations with low-data transfer capacity associations. Since the endorser can keep going for some time without an update, this gives an answer that is lower in cost than different strategies while yet taking care of the necessities.

Depiction replication likewise has the additional advantage of being the main replication type in which the reproduced tables are not expected to have an essential key. Depiction replication works by perusing the distributed data set and making records in the functioning organizer on the wholesaler. These records are called depiction documents and contain the information from the distributed data set as well as some extra data that will assist with making the underlying duplicate on the membership server. [5]

Depiction replication is many times utilized while expecting to peruse information, for example, cost records, online inventories, or information for choice help, where the latest

information is not fundamental, and the information is utilized as perused as it were.

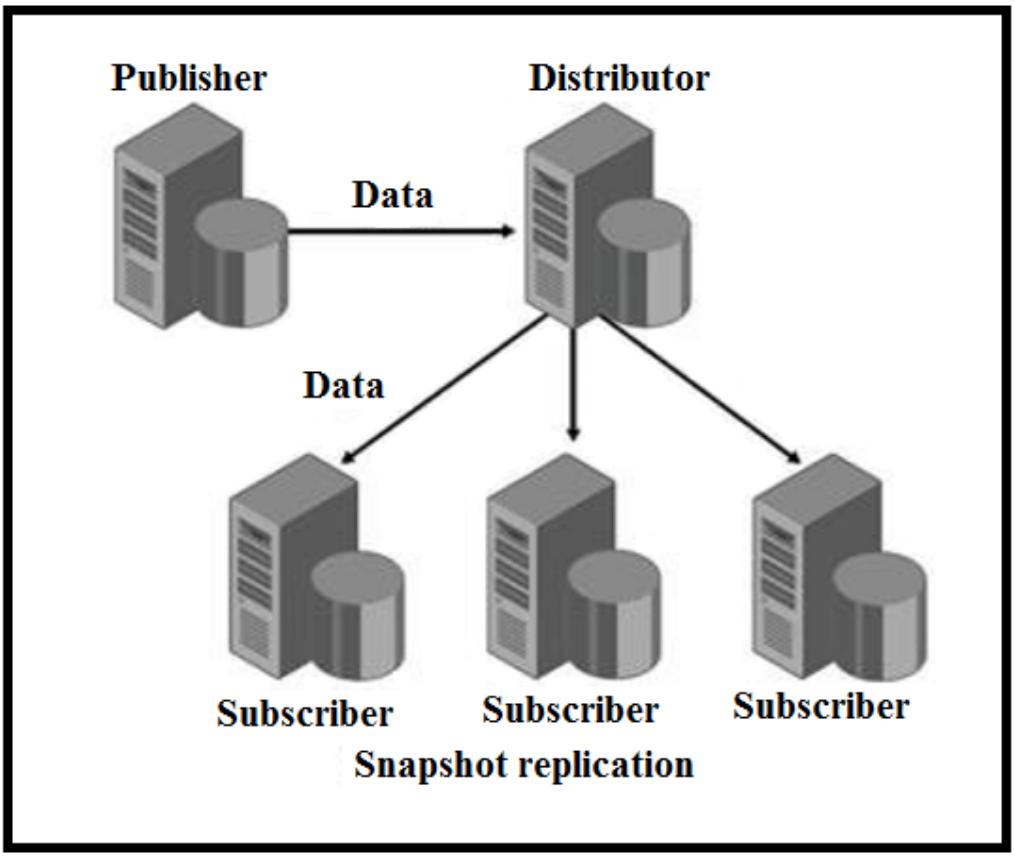

Figure 2.1 Snapshot Replication

Snapshot replication is helpful when:

- Data is mostly static and does not change often.
- It is acceptable to have copies of data that are out of date for a period of time.
- Replicating small volumes of data in which an entire refresh of the data is reasonable [5].

2.9.2 Merging Replication

Merge replication is the method involved with conveying information from Publisher to Subscribers, permitting the Publisher and Subscribers to make refreshes while associated or detached, and afterward combining the updates between locales when they are associated [7].

Combine replication permits different locales to work independently and sometime in the not too distant future consolidation refreshes into a solitary, uniform outcome. The underlying

preview is applied to Subscribers, and afterward changes are followed to distribute information at the Publisher and at the Subscribers. The information is synchronized between servers constantly, at a booked time, or on request. Since refreshes are made at more than one server, similar information might have been refreshed by the Publisher or by more than one Subscriber. Accordingly, clashes can happen when updates are combined [5].

Combine replication incorporates default and custom decisions for compromise that design a consolidation distribution. At the point when a contention happens, a resolver is conjured by the Merge Agent and figures out which information will be acknowledged and engendered to different destinations.

Combine Replication is useful when:

- Numerous Subscribers need to refresh information at different times and engender those changes to the Publisher and to different Subscribers.
- Endorsers need to get information, make changes disconnected, and later synchronize changes with the Publisher and different Subscribers.
- Client doesn't expect many contentions when information is refreshed at numerous destinations (in light of the fact that the information is sifted into segments and afterward distributed to various Subscribers or due to the purposes of client's application). Nonetheless, assuming struggles do happen, infringement of ACID properties are satisfactory [3].

2.9.3 Transactional Replication

In what could be viewed as something contrary to depiction replication, conditional replication works by sending changes to the endorser as they occur. For instance, SQL Server processes all activities inside the information base utilizing Transact-SQL explanations. Each finished assertion is known as an exchange.

In conditional replication, each serious exchange is imitated to the endorser as it happens. The replication interaction will aggregate exchanges and send them at planned spans or communicate all changes as they happen. This kind of replication has a lower level of inertness and higher data transmission associations [7].

Conditional replication requires a nonstop and solid association, on the ground that the Transaction Log will develop rapidly in the event that the server can't interface for replication and

could become unmanageable. Value-based replication starts with a depiction that sets up the underlying duplicate. That duplicate is afterwards refreshed by the replicated exchanges. How frequently to refresh the depiction, or decide not to refresh the preview after the principal duplicate [5].

When the underlying depiction has been duplicated, conditional replication utilizes the Log Reader specialist to peruse the Transaction Log of the distributed data set and stores new exchanges in the appropriation data set. The dispersion specialist then, at that point, moves the exchanges from the distributer to the endorser.

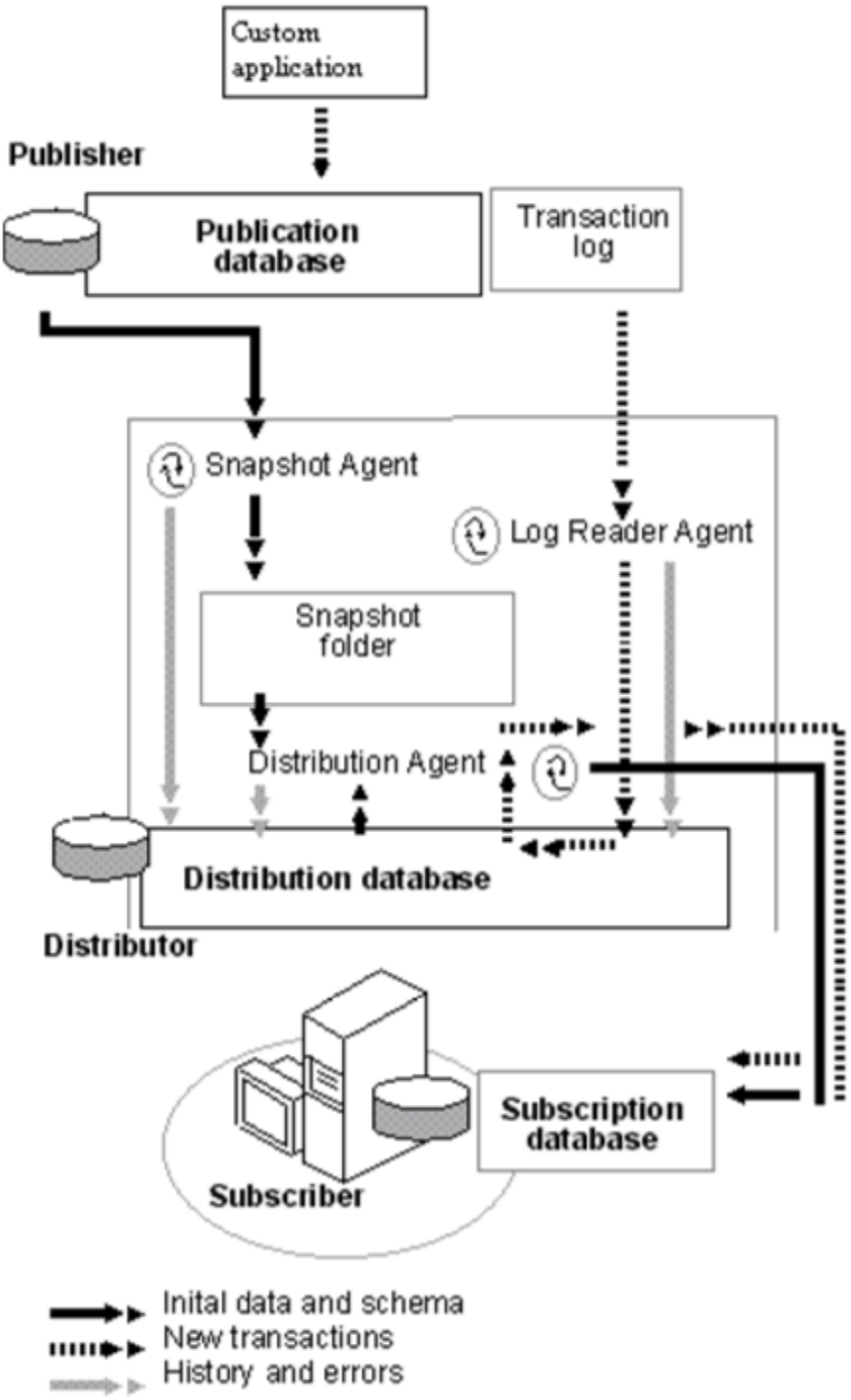

Figure 2.2 Transactional Replication

Transactional replication with updating subscribers: A branch-off of standard value-based replication, this technique for replication essentially works the same way, yet adds to supporters the capacity to refresh information. At the point when a supporter rolls out an improvement to information locally, SQL Server utilizes the Microsoft Distributed Transaction

Coordinator (MSDTC), a part included with SQL Server, to execute a similar exchange on the distributer. This cycle takes into account replication situations in which the distributed information is viewed as perused just more often than not yet can be changed at the endorser once in a while if necessary.

Value-based replication with refreshing supporters requires an extremely durable and dependable association of medium to high data transmission. [5] Subscribers are dependably and additionally oftentimes associated with the Publisher. [6]

2.10 Transactions

Transactions are sequences of data operations terminated by one control operation. Data operations are read and write, control operations are commit and abort [3].

2.11 Distributed Transaction Management

The goal of transaction is to ensure that all of the objects managed by a server remain in a consistent state in the presence of server crashes. The server must guarantee that both transactions are carried out and the results recorded in permanent storage, or in the case of crashes, the effects are completely erased. The object that can be recovered after the server crashes is called recoverable object.

A client transaction becomes distributed if it invokes operations in several different servers. There are two different types of distributed transactions: Flat transaction

- Flat transaction
- Nested transaction

In a flat transaction, a client makes requests to more than one server. For example, in figure 1, transaction T is a flat transaction that invokes operations on objects in servers X, Y and Z. A flat client transaction completes each of its requests before going on to the next one. Therefore, each transaction accesses servers' objects sequential [3].

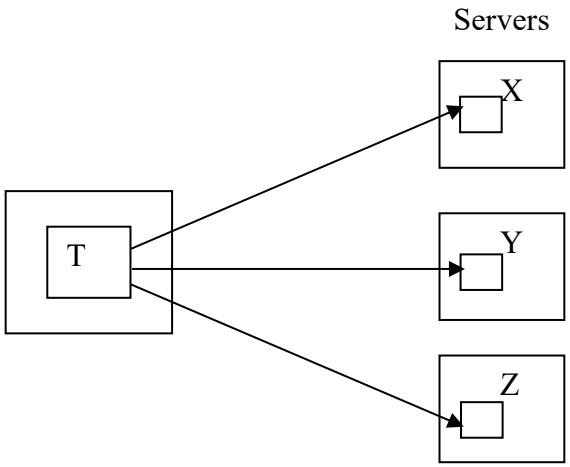

Figure 2.3 Flat Distributed Transaction

In a nested transaction, the top-level transaction can open sub-transactions and each sub-transaction can open further sub-transactions down to any depth of nesting. [8]

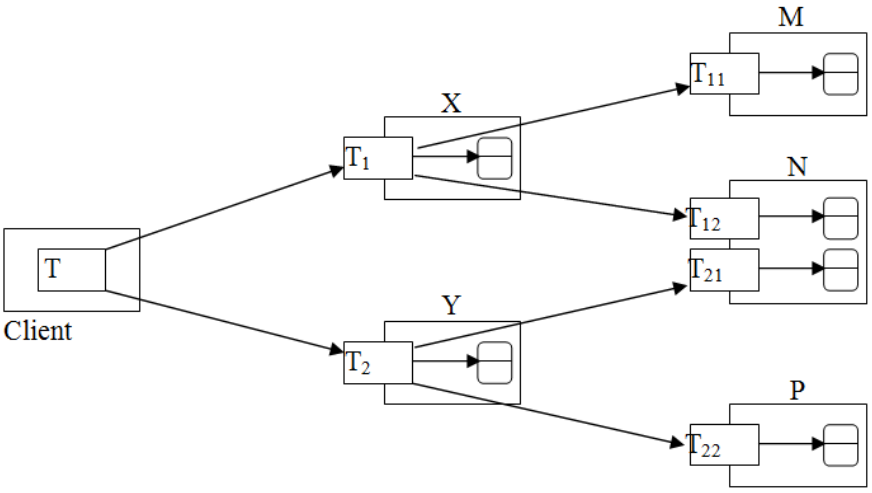

Figure 2.4 Nested Distributed Transactions

Figure 2.4 shows a client's transaction T that opens two sub-transactions T1 and T2, which access objects at server X and Y. The sub-transactions T1 and T2 open further sub-transactions T11, T12, T21 and T22, which access objects at servers M, N and P. In the nested case, sub-transactions at the same level can run concurrently, so T1 and T2 are concurrent, and they can run in parallel. The four sub-transactions T11, T12, T21 and T22 also run concurrently [8].

CHAPTER 3
ACTIVE AND PASSIVE REPLICATIONS USING MULTICAST

In a distributed system data is store is over different computers in a network. Therefore, we need to make sure that data is readily available for the users. Availability of the data is an important factor accomplished by data replication.

3.1 Replication

In current circulated frameworks, replication in registering includes dividing data to guarantee consistency among repetitive assets, for example, programming or equipment parts to further develop execution and accessibility, dependability, adaptation to non-critical failure: disappointment can be stowed away from clients and applications on the off chance that they can get information administrations from an indistinguishable imitation; replication can further develop execution by scaling the quantity of reproductions with request and by offering close by duplicates to administrations disseminated over a wide region[2].

A major test with replication is to keep up with information consistency among copies. In a replication framework, the worth of each legitimate thing is put away in at least one actual information things, alluded to as its duplicates. Each read or composes procedure on a sensible information thing should be planned to comparing procedure on actual duplicates. Precisely when and how these mappings are completed decides the consistency ensures given by the framework and the expense of replication.

Reliability: Replication removes a single point of failure and allows consensus protocols to deal with corrupted data (majority voting, etc.)

Availability: Clients can get the help with sensible reaction time, for however much time as could be expected.

Performance: Location transparency is difficult to achieve in a distributed environment. Local accesses are fast, remote accesses are slow. If everything is local, then all accesses should be fast.

Fault Tolerance: Failure resilience is also difficult to achieve. If a site fails, the data becomes unavailable. By keeping several copies of the data at different sites, single site failures should not affect the overall availability [4].

Correspondence between various framework parts (clients and imitations) happens by trading messages. Appropriated frameworks recognize the coordinated and the offbeat framework model.

In the coordinated model there is a realized bound on the overall cycle speed and on the message transmission delay, while no such limits exist in the offbeat model. The key distinction is that the coordinated framework permits right accident identification, while the nonconcurrent framework doesn't (i.e., in an offbeat framework, when some cycle p feels that some other cycle q has crashed, q could as a matter of fact not have crashed). Mistaken crash location makes the improvement of replication calculation more troublesome. A significant part of the intricacy can be concealed in the supposed gathering correspondence natives [4].

Information bases are not worried by the principal distinctions among coordinated and non-concurrent frameworks for the accompanying explanation: data sets acknowledge to live with hindering conventions (a convention is supposed to impede on the off chance that the accident of some cycle might keep the convention from ending). Impeding convention is much easier to plan than non-hindering conventions in light of the coordinated model. Dispersed frameworks normally search for non-obstructing conventions. This reflects one more principal contrast between conveyed frameworks and data set replication conventions. It has been demonstrated the way that the specification of each and every issue can be disintegrated into security and liveness properties.

Information based conventions do not treat liveness issues officially, as a feature of the convention specification. For sure, the properties guaranteed by exchanges (Atomicity, Consistency, Isolation, Durability) are all wellbeing properties. In any case, since data sets acknowledge to live with hindering conventions, liveness isn't an issue. With the end goal of this paper, we focus on security properties. At long last, data set replication conventions might concede, at times, administrator mediation to tackle strange cases, similar to the disappointment of a server and the arrangement of another (a method for dodging impeding). This is normally not done in dispersed framework conventions, where the substitution of a reproduction by one more is coordinated into the convention (non-hindering conventions) [1].

3.2 Functional Model

A replication convention can be portrayed utilizing five nonexclusive stages. These stages address significant stages in the convention and will be utilized to describe the various methodologies. Some replication methods might skirt a few stages, request them in an alternate way, repeat over some of them, or consolidation them into a less difficult succession. Subsequently, the conventions can measure up by the manner in which they carry out every last one of the stages and how they join the various stages. In such manner, a theoretical replication convention can be depicted as a grouping of the accompanying five stages (as shown in Figure 3.1).

Request (RE): the client submits an operation to one (or more) replicas.

1. **Server coordination (SC):** the replica servers coordinate with each other to synchronize the execution of the operation.
2. **Execution (EX):** the operation is executed on the replica servers.
3. **Agreement coordination (AC):** the replica servers agree on the result of the execution.
4. **Response (END):** the outcome of the operation is transmitted back to the client [8].

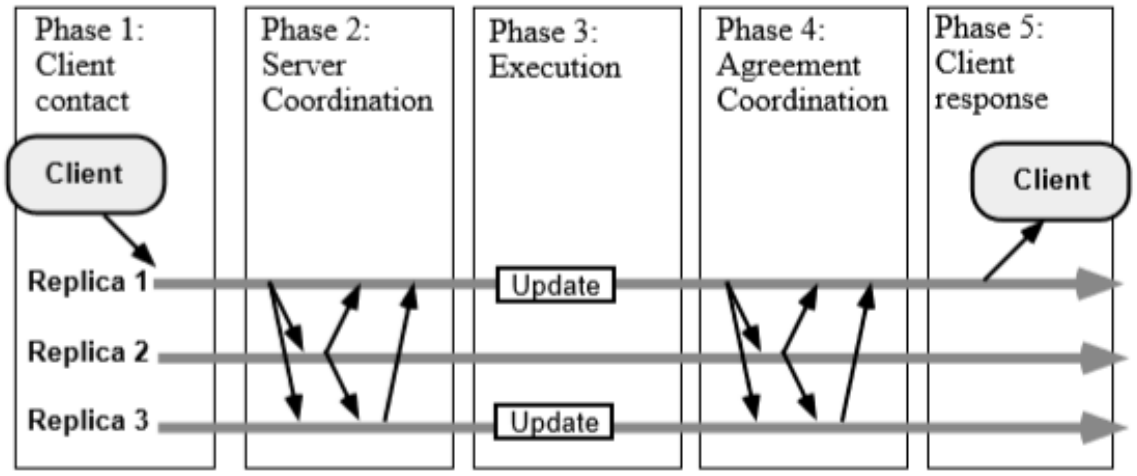

Figure 3.1 Functional model with the five phases

This useful model addresses the fundamental stages of replication: accommodation of an activity, coordination among the imitations (e.g., to arrange simultaneous tasks), execution of the activity, further coordination among the reproductions (e.g., to ensure atomicity), and reaction to the client. The distinctions between conventions emerge because of the various methodologies utilized in each stage which, now and again, deter the requirement for another stage (e.g., when messages are requested in view of a nuclear transmission crude, the understanding coordination

stage isn't required since it is as of now proceeded as a component of the cycle or requesting the messages).

Exchange can be a solitary perused or composed activity, a more complicated activity with numerous boundaries, or a summon on a technique. Albeit prohibitive from the beginning, this model is taken on by some information base sellers, to deal with web archives and put away strategies.

Request Phase: During the solicitation stage, a client presents an activity to the framework. This should be possible in two ways: the client can straightforwardly send the activity to all copies, or the client can send the activity to one imitation which wills them send the activity to all others as a component of the server coordination stage.

This differentiation, albeit evidently basic, as of now presents some significant contrasts among data sets and dispersed frameworks. In data sets, clients never contact all imitations, and consistently send the activity to one duplicate. The explanation is extremely basic: replication ought to be straightforward to the client. Having the option to send an activity to all imitations will suggest the client knows about the information area, mapping, and appropriation which isn't viable for any data set of normal size. This is information characteristically attached to the data set hubs, subsequently, client should constantly present the activity to one hub which will then, at that point, send it to all others. In disseminated frameworks, be that as it may, a reasonable qualification is made between replication strategies relying upon whether the client sends the activity straightforwardly to all duplicates (for example dynamic replication) or to one duplicate (for example latent replication) [8].

It very well may be contended that in the two cases, the solicitation components should be visible as reaching an intermediary (a data set hub in one case, or a correspondence module in the other), in which case there are no significant contrasts between the two methodologies. Theoretically this is valid. Essentially, it's anything but an exceptionally supportive deliberation in light of its suggestions as it will be examined underneath when the various conventions are looked at. For the second being, note that disseminated frameworks manage processes while data set manage social patterns. A rundown of cycles is more straightforward to deal with that an information base blueprint, i.e., a correspondence module can be anticipated to have the option to deal with a rundown of cycles, yet it isn't reasonable to expect it can deal with a data set diagram.

Specifically, data set replication requires understanding the activity that will be performed while in circulated frameworks, activity semantics typically assume no part.

At long last, appropriated frameworks recognize deterministic and non-deterministic copy conduct. Deterministic imitation way of behaving accepts that when the solicitation gave similar activities in similar request, copies will create similar outcomes. Such a supposition that is undeniably challenging to make in a data set[7].

Consequently, in the event that the various imitations need to convey in any case to settle on an outcome, they can trade the genuine effort. By moving the weight of transmission, the solicitation to the server, the rationale fundamental at the client side is extraordinarily simplified at the cost of (hypothetically) diminishing adaptation to non-critical failure. In the event that adaptation to internal failure is vital, a reinforcement framework can be utilized, however this is absolutely straightforward to the client.

Server Coordination Phase: During the server coordination stage, the various imitations attempt to find a request where the activities should be performed. Here conventions contrast the most with regards to requesting methodologies, requesting components, and rightness rules [8].

As far as requesting methodologies, information bases request activities as per information conditions. That is, all activities should have similar information conditions at all imitations. It is a direct result of this reason that activity semantics assume a significant part in data set replication: an activity that main peruses an information thing isn't equivalent to an activity that modifies that information thing since the information conditions presented are not similar in the two cases. Assuming there are no immediate or aberrant conditions between two tasks, they needn't bother with to be requested on the grounds that the request doesn't make any difference [2].

Disseminated frameworks, then again, are generally founded on exceptionally severe ideas of requesting. From causality, which depends on possible conditions without taking a gander at the activity semantics, to add up to arrange (either causal or not) in which all tasks are requested paying little heed to what they are.

Execution Phase: The execution stage addresses the genuine performing of the activity. It doesn't present numerous distinctions between conventions, yet it is a decent sign of how each approach treats and circulates the tasks. This stage just addresses the genuine execution of the activity, the applying of the update is normally finished in the Agreement Coordination Phase, despite the fact that applying the update to different duplicates might be done byre-executing the tasks [8].

Agreement Coordination Phase: During this stage, the various imitations ensure that they all do the same thing. This stage is fascinating in light of the fact that it raises a portion of the major distinctions between conventions. In data sets, this stage normally compares to a Two-Phase Commit Protocol (2PC) during which it is concluded whether the activity will be committed or cut off.

This stage is important in data set, the Server Coordination stage takes care just of requesting activities. When the requesting has been settled upon, the reproductions need to guarantee everyone consents to really committing the activity. Note that having the option to arrange the tasks doesn't be guaranteed to mean the activity will succeed. In a data set, there can be many justifications for why an activity prevails at one site and not at another (heap, consistency requirements, and communications with neighborhood tasks).

This is a central contrast with dispersed frameworks where when an activity has been effectively requested (in the Server Coordinator stage) it will be conveyed (i.e., "performed") and there is compelling reason need to do any further checking.

Client Response Phase: The client reaction gradually ease addresses the second in time when the client gets a reaction from the framework. There are two prospects: either the reaction is sent solely after all that has been settled and the activity has been executed, or the reaction is sent immediately and the proliferation of changes and coordination among all reproductions is done subsequently. On account of data sets, this qualification prompts 1) the supposed anxious or coordinated (no reaction until all that has been finished) and 2) apathetic or offbeat (quick reaction, engendering of changes is done subsequently) conventions [10].

In the appropriated frameworks case, the reaction happens solely after the convention has been executed and no errors might emerge. The client reaction stage is of expanding significance given the multiplication of utilizations for portable clients, where a duplicate isn't generally associated with the remainder of the framework, and it doesn't check out to hold on until refreshes occur to allow the client to see the progressions made.

3.3 Active Replication

Replication is a strategy for improving administrations. The inspirations for replication are to work on a help's presentation, to expand its accessibility, or to make it issue lenient. Replication comprises of at least two imitations. Imitations are actual items, each put away at a

solitary PC. Copies are held by unmistakable reproduction chiefs. Copy chiefs are parts that contain the imitations on a given PC and perform tasks up on them straightforwardly.

There are two approaches for fault-tolerant service:

1. Passive (primary-backup) replication
2. Active replication

Active replication, also called the state machine approach, is a non-incorporated replication strategy. Its key idea is that all reproductions get and deal with a similar grouping of client demands [6].

Consistency is ensured by expecting to be that, when given similar contribution to similar request, imitations will create a similar result. This presumption suggests that servers cycle demands in a deterministic manner. Clients don't reach one specific server, yet address servers collectively. For servers to get a similar contribution to a similar request, client solicitations can be proliferated to servers utilizing an Atomic Broadcast. More fragile correspondence natives can likewise be utilized assuming semantic data about the activity is known (e.g., two demands that drive don't need to be conveyed at all servers in a similar request).

The primary benefit of dynamic replication is its straightforwardness (e.g., same code all over the place) and disappointment straightforwardness. Disappointments are completely stowed away from the clients, since on the off chance that a reproduction falls flat; the solicitations are as yet handled by different copies. The determinism requirement is the significant disadvantage of this methodology. Albeit one could likewise contend that having all the handling done on all reproductions consumes an excess of asset. Notice notwithstanding, that the other option, or at least, handling a solicitation at only one copy and communicating the state changes to the others (see next segment), at times might be significantly more complicated and costly than basically executing the summon on all destinations[9].

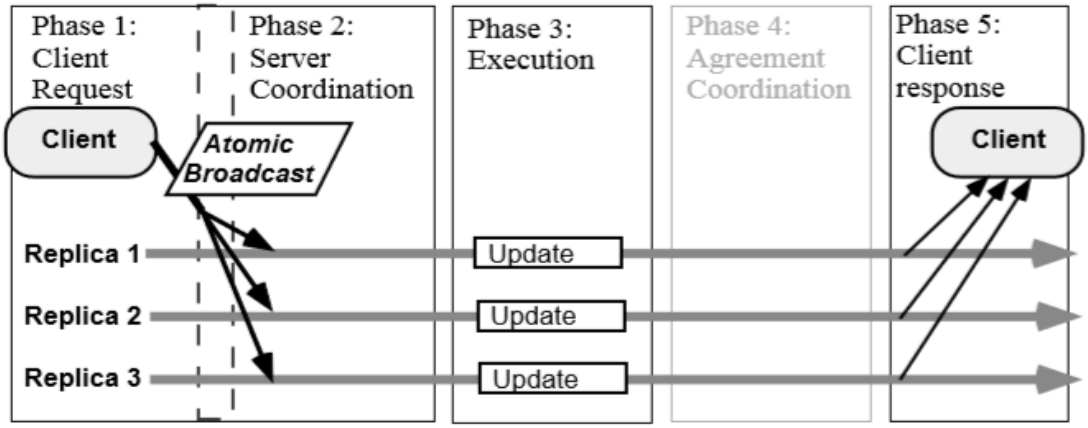

Figure 3.2 Active replication

Figure 3.2 depicts the active replication technique using an Atomic Broadcast as communication primitive. In active replication, phases Request (RE) and Server Coordination (SC) are merged and phase Agreement Coordination (AC) is not used.

The following steps are involved in the processing of an update request in the Active Replication, according to our functional model.

1. The client sends the request to the servers using an Atomic Broadcast.
2. Server coordination is given by the total order property of the Atomic Broadcast.
3. All replicas execute the request in the order they are delivered.
4. No coordination is necessary, as all replica process the same request in the same order. Because replicas are deterministic, they all produce the same results.
5. All replicas send back their result to the client, and the client typically only waits for the first answer (the others are ignored) [6].

In this system, active replication will be used for fault tolerant. General model of replica management is shown in Figure 3.3.

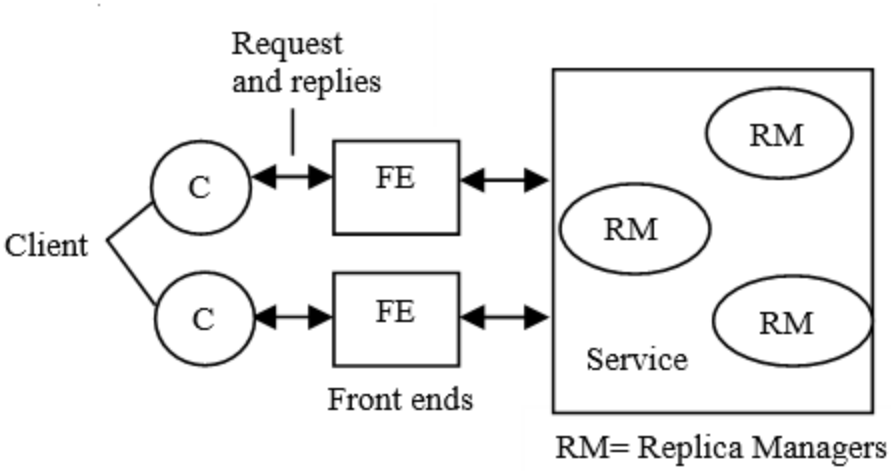

Figure 3.3 General Model of Replica Management

3.4 Passive Replication

The basic principle of passive replication, also called Primary Backup replication, is that clients send their solicitations to an essential, which executes the solicitations and sends update messages to the reinforcements (as shown in Figure 3.4). The reinforcements do not execute the conjuring, yet apply the progressions delivered by the summon execution at the essential (i.e., refreshes). By doing this, no determinism requirement is fundamental on the execution of summons [5].

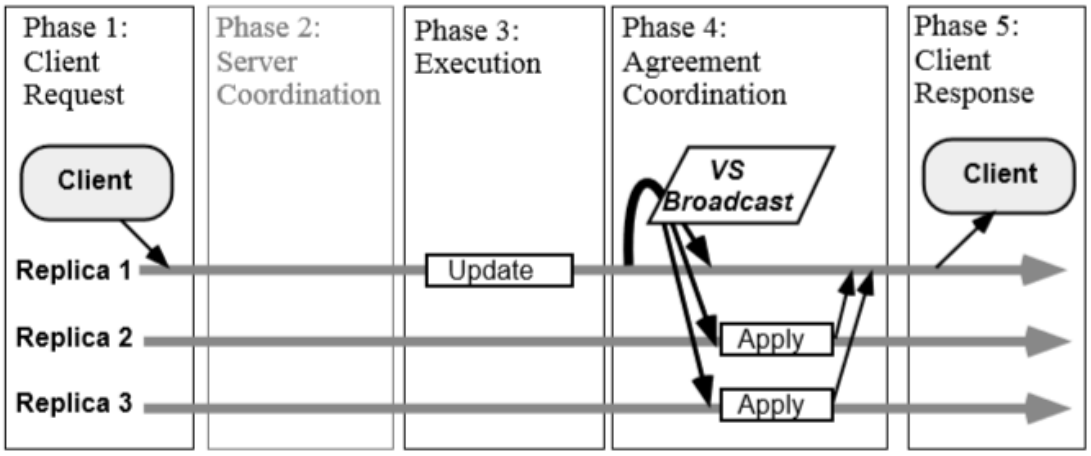

Figure 3.4 Passive replication

Correspondence between the essential and the reinforcements needs to ensure that updates are handled in a similar request, which is the situation on the off chance that essential reinforcement correspondence depends on FIFO channels. Nonetheless, a main FIFO channel

isn't sufficient to guarantee right execution in the event of disappointment of the essential. For instance, consider that the essential bombs before all reinforcements get the updates for a specific solicitation, and another reproduction takes over as another essential. Some system needs to guarantee that updates sent by the new essential will be "appropriately" requested as to the updates sent by the broken essential. VSCAST is a component that ensures these imperatives and can for the most part be utilized to execute the essential reinforcement replication method [2].

Latent replication can endure non-deterministic servers (e.g., multi-strung servers) and uses little handling power when contrasted with other replication procedures. Notwithstanding, inactive replication experiences a high reconfiguration cost when the essential falls flat.
The five strides of the proposed system are as follows:

1. The client sends the solicitation to the essential.
2. There is no underlying coordination.
3. The primary executes the solicitation.
4. The primary directions with different reproductions by sending the update data to the reinforcements.
5. The primary sends the solution to the client[8]. The passive model for fault tolerance is shown in Figure 3.5.

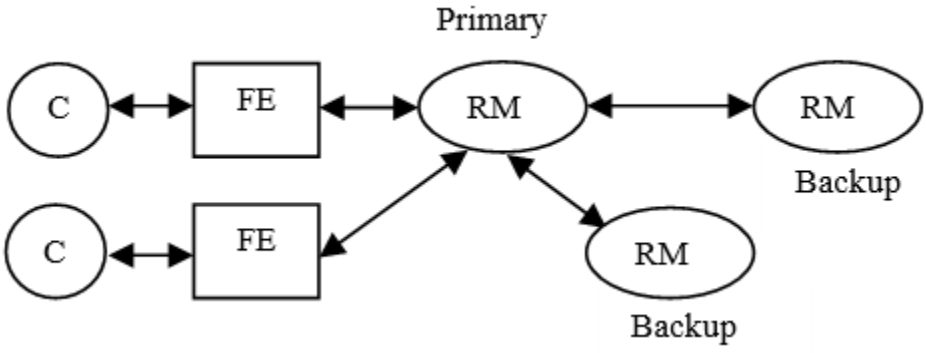

Figure 3.5 General Model of Replica Management (In Passive)

In this model, there is at any one time, a solitary essential imitation chief and at least one optional copy supervisors - 'reinforcements' or 'slaves'. In the unadulterated type of the model, front closures discuss just with the essential reproduction chief to get the help. The essential copy administrator executes the tasks and sends duplicates of the refreshed information to the

reinforcements. In the event that the essential falls flat, one of the reinforcements is elevated to go about as the primary.

CHAPTER 4
THE SYSTEM DESIGN AND IMPLEMENTATION

This system implemented a reliable furniture ordering system using backup replica servers. Primarily, the furniture ordering web system used the main server and two backup servers. Every transaction (such as ordering transaction from the clients or data entry transaction of the admin) is made at the primary server of the system. Similarly, the backup servers of the system must be executing the processing transaction to maintain the data consistency between primary server and backup replica servers.

When the backup replicas are not idle to execute the processing transaction to maintain the data consistency between primary server and backup replica servers, only the primary server process the requested transaction and then the committed results are propagated to replicas and applied to them. So, this system controls not only the data consistency and reliability but also the avoidance for the waiting to the busy replicas by the used of LLFT. The detailed process flow of the system is shown in Figure 4.1.

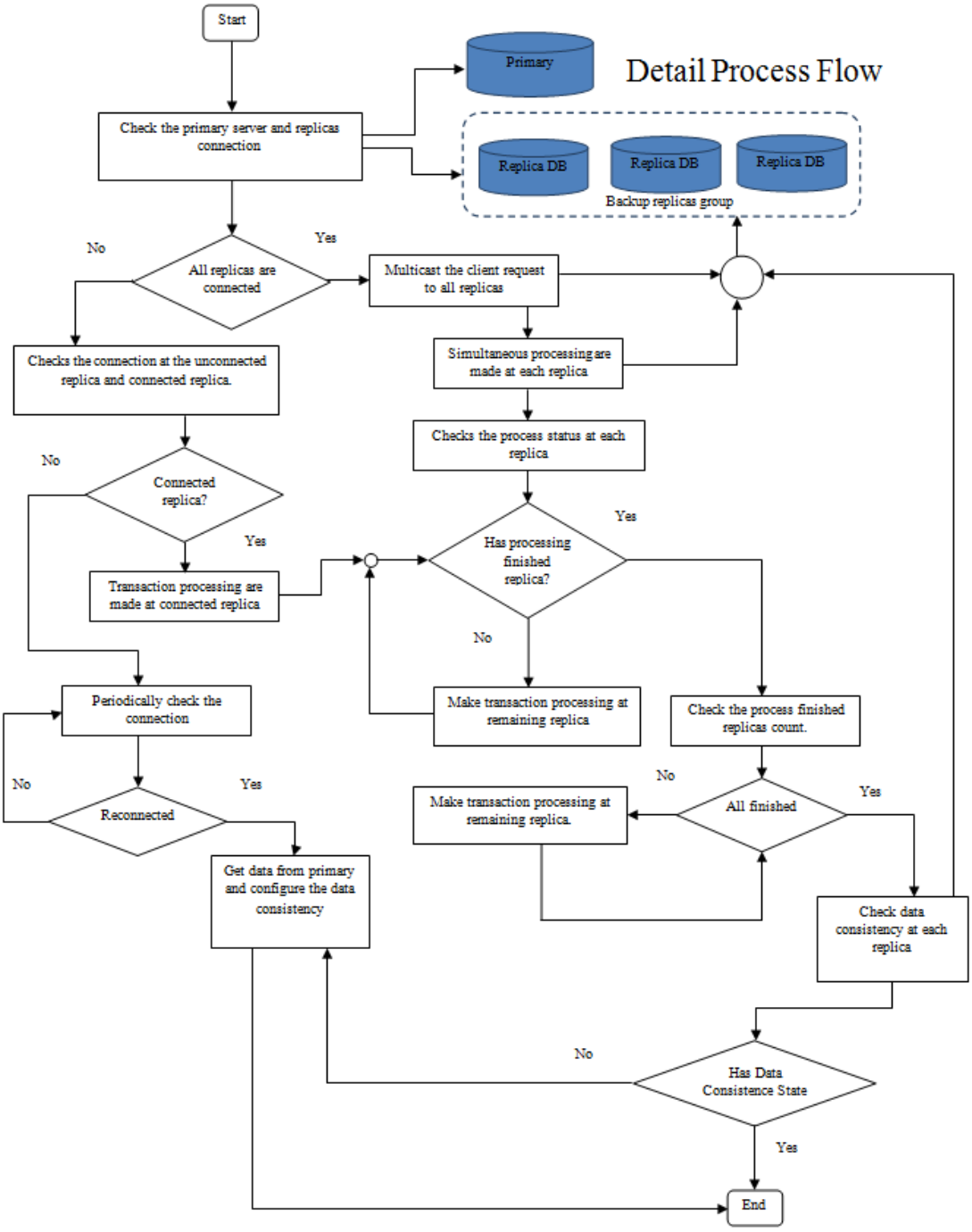

Figure 4.1: The System Flow

4.1. Example Operation of Semi- active and Semi-passive in Proposed System

Handling the solicitation from the first client refreshes an information thing. Handling the solicitation from the subsequent client refreshes similar information thing, where the collaboration between the handling of the two solicitations is non-deterministic. The solicitation handling finishes, and the essential sends answers to the clients. The essential then, at that point, bombs before it sends its updates to the reinforcements. The handling of the solicitations from the two clients is rehashed at the new essential.

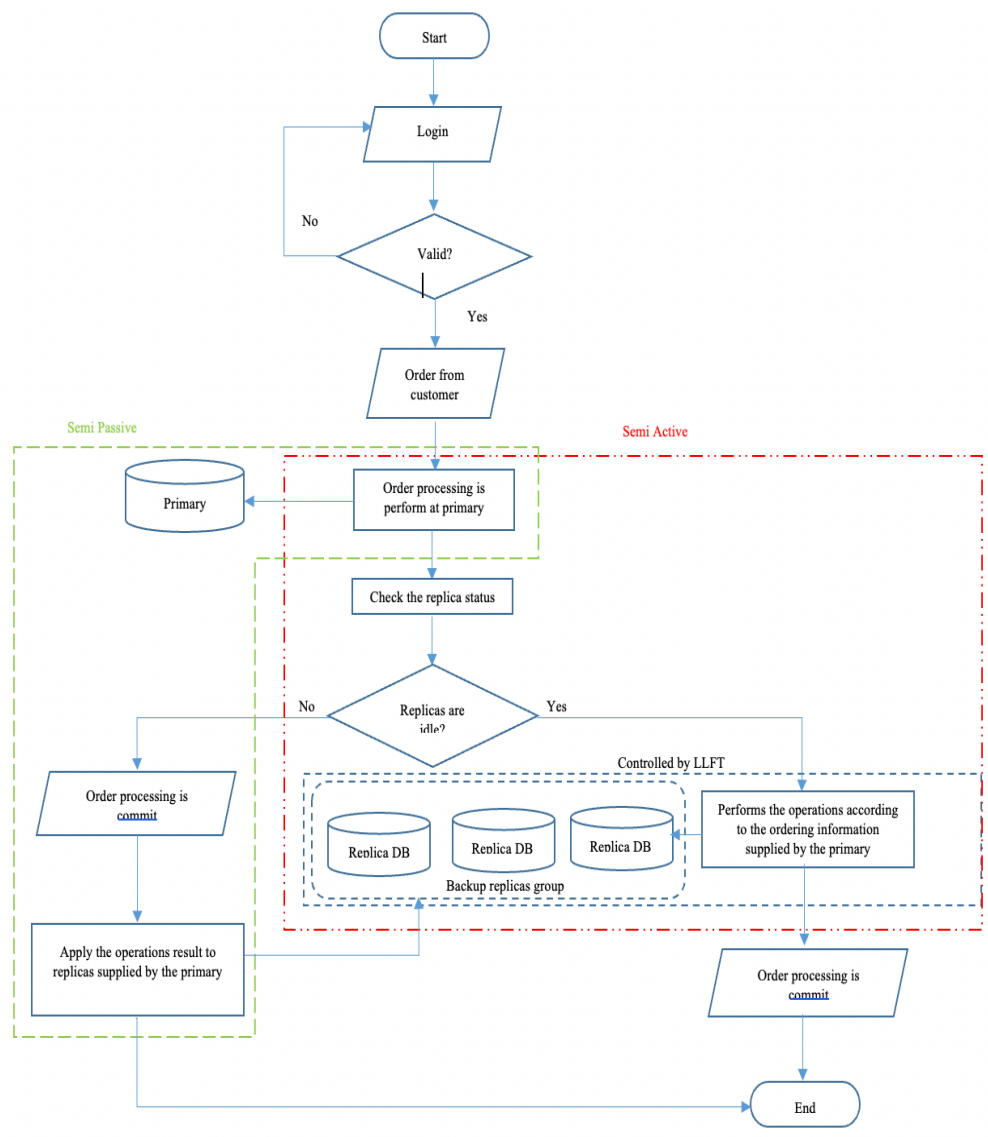

Figure 4.2: The Detail Process Flow of LLFT in Proposed System

Be that as it may, the non-deterministic cooperation between the handling of the two solicitations are shipped off the clients. The handling of the solicitations at the new essential should

rehash similar non-deterministic cooperation, on the off chance that the right outcomes are to be acquired. In the event that a gathering of reproductions becomes divided, LLFT guarantees that only one part of the parcel, alluded to as the essential part. Inside the essential part, LLFT keeps up with virtual synchrony, i.e., assuming the essential comes up short, the new essential should progress to the condition of the old essential, and the state known to the remote gatherings of its associations, before the old essential fizzled. The cycles of different parts could end tasks and must reapply for admission to the participation. Care should be taken to recuperate those activities and to reestablish consistency. LLFT guarantees to proceed with the activity and abstain from hindering during apportioning. The detailed cycle stream of LLFT activity is displayed in Figure 4.2. The UI plans of the shortcoming control status are displayed in the following Figure 4.3 and Figure 4.4.

Figure 4.3: Dashboard –Primary Down (Admin Site)

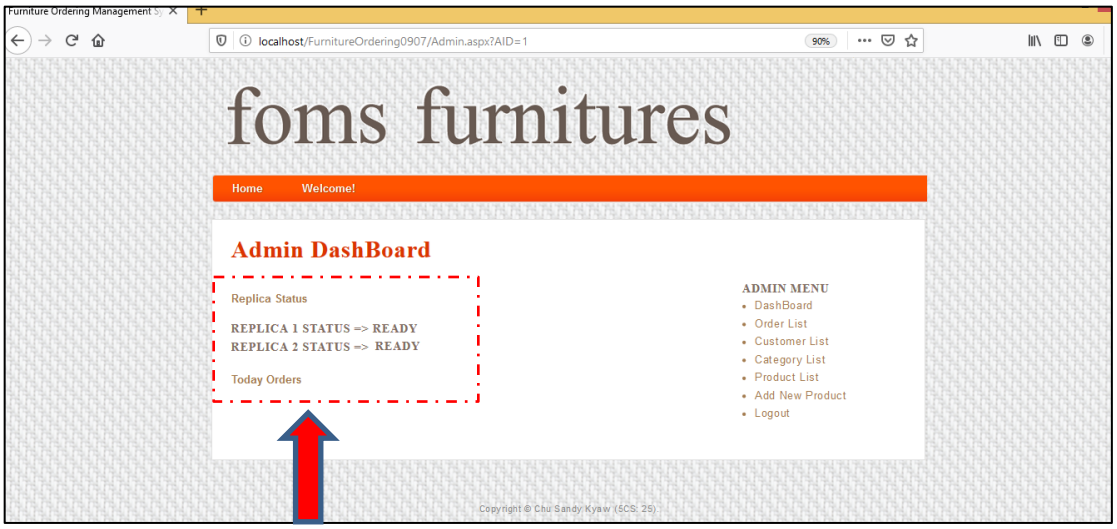

Figure 4.4: Dashboard –Two Replicas – Ready (Admin Site)

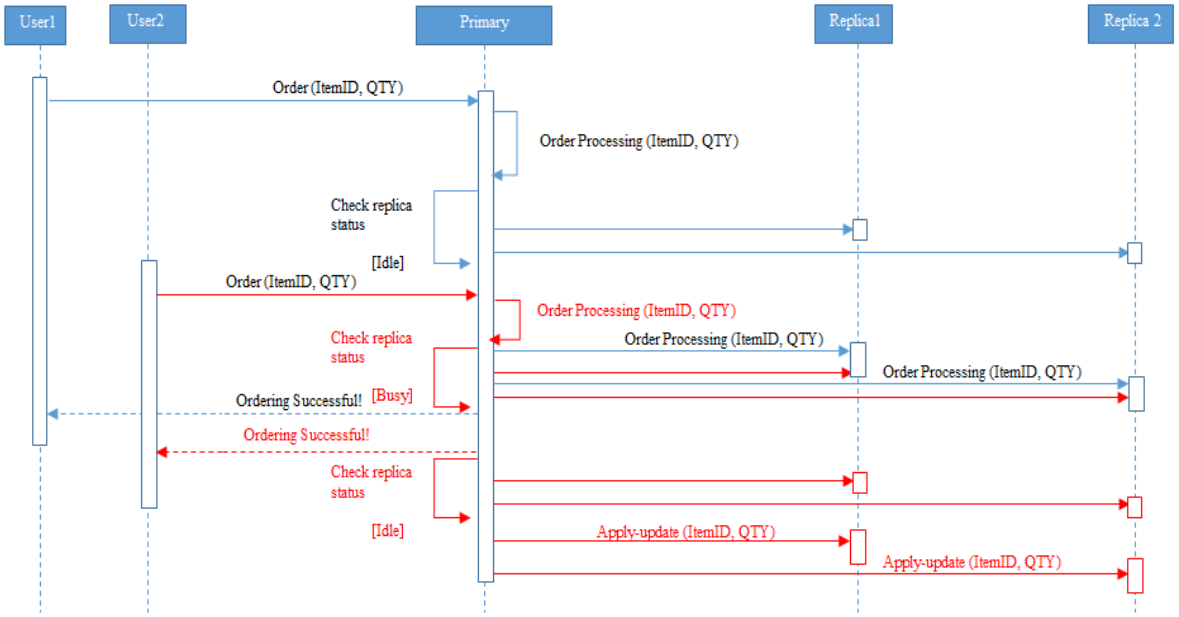

Figure 4.5: Sequence Diagram of System

The above Figure 4.5 describes another comprehension for data update distribution to the two remaining replicas to control the low latency fault tolerance. Due to the fact that the data update are timely distributed to replicas, the system can continue without delaying the processing task in case of primary server downing.

4.2. The Proposed Low Latency Fault Tolerance Algorithm

BEGIN

Step1: Accept the request from clients;

Step2: Requests processing are made at primary;

```
If (The request processing completes)
{
        GOTO: Step3;
}
Else
{
        Repeat: Step2;
}
End If
```

Step3: The primary sends replies to the clients;

Step4: Check the primary status;

```
If (Status = = OK)
{
        Check the replicas status;
        For (int r=1; r <= number of replica; r++)
        {
                If (replica [r] status = = OK)
                {
                        Primary sends its updates to the backup/replica [r].
                }
                Else If (replica [r] status = = Busy)
                {
                        Primary passively sends its updates to the backup/replica [r].
                }
        }

}
```

Else If (The primary then fails before it sends its updates to the backups)

{

Revoke the new primary from the replicas/backups;

Repeat: Step2;

}

End If

Step 5: Check the group of replicas;

If (a group of replicas becomes network disconnected)

{

If (network disconnected component = = primary component)

{

LLFT allows continued operation and avoid blocking during network disconnected.

}

Else If (network disconnected component = = other components)

{

Might terminate operations and must reapply for admission to the membership;

}

End If

}

End If

END

4.3. Implementation of the System

This section describes the user interface of the system implementation. This system is developed for the dealing process of furniture ordering system. In this system, the transaction processing recovery is made between the dealers and main server of the system. This system is implemented by using ASP.Net Language on IIS Web Server.

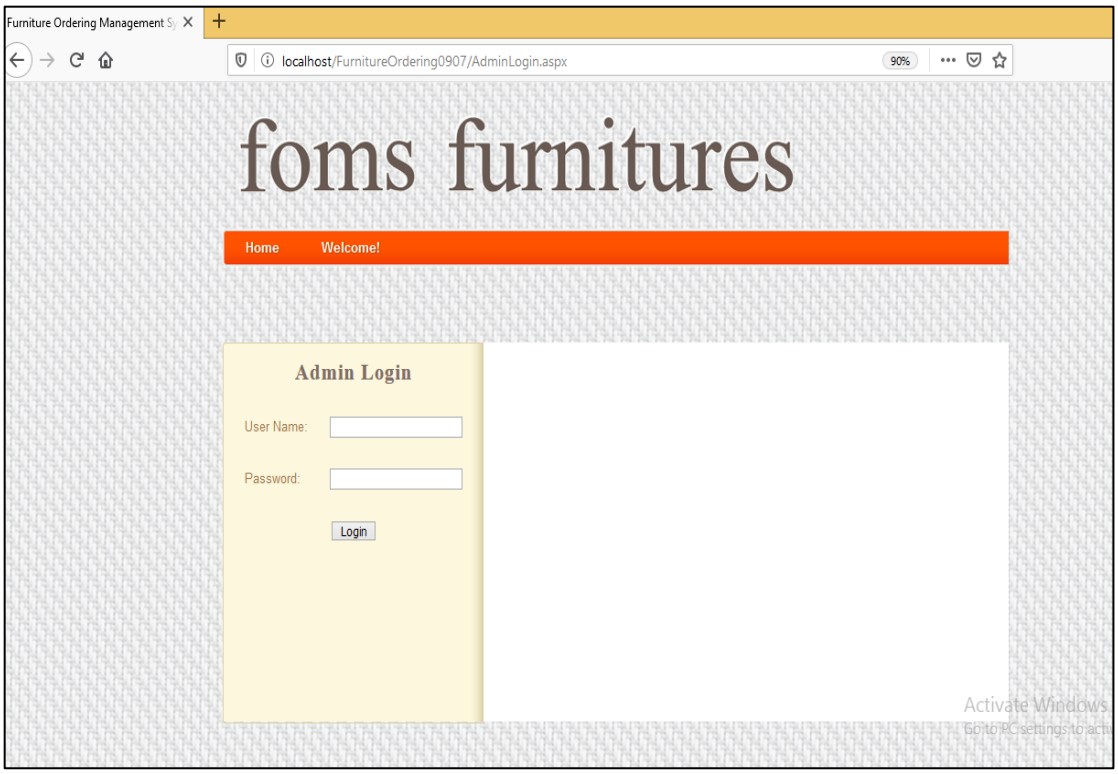

Figure 4.6: Admin Login

The system has two level users: the login user for server side and the login user for client side. Not only login at the server side but also login at the client side, the system allows only the registered users. The Admin Login page is shown in Figure 4.6 and the customer login page is also shown in Figure 4.7.

Figure 4.7: Customer Login Page

4.3.1. Adding New Product and New Category at the Admin Site

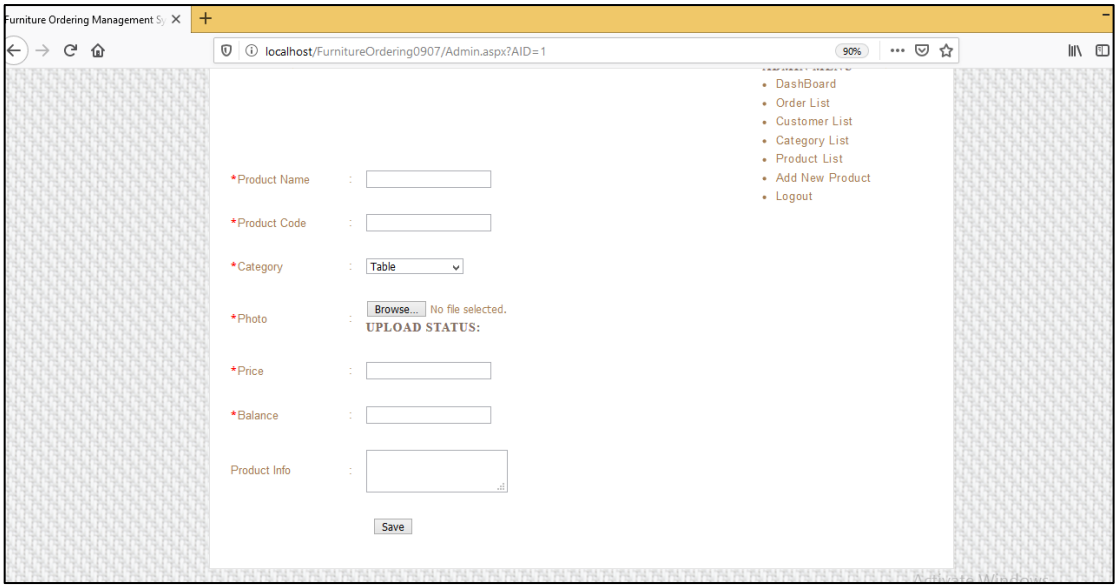

Figure 4.8: Add New Product (Admin Site)

Figure 4.8 shows the new product adding page of the system. In this system, the dealing furniture products are added to the web page by the admin. In the product detail description, the admin must add the Product name, Product code, Category (such as table, chair, cupboard, lift chair, bean bag, bed and so on), sample photo, retail price, balance of product and detail information. The admin can add the new product category to the system by using the following web page of Figure 4.9.

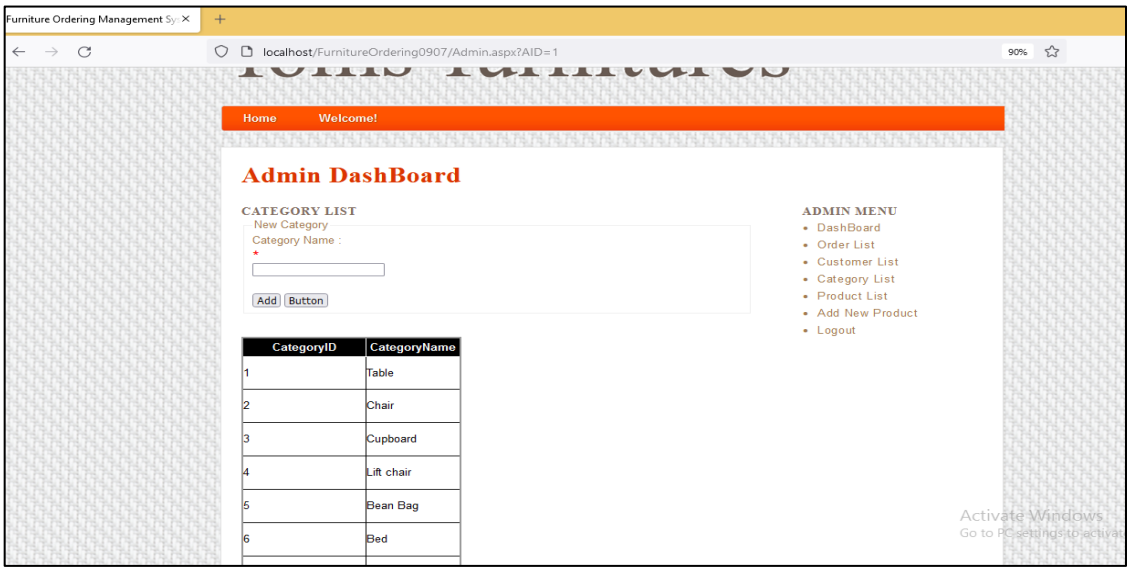

Figure 4.9: Category List (Admin Site)

The product list added by the admin is shown in figure 4.10. The product list helps to the admin side to easily check the dealing products balance information, editing the change of prices and so on. The admin side also includes the following tags (Such as Dashboard, Order List, Customer List, Category List, Product List, Add New Product and Logout Tags) for ease and fast of access.

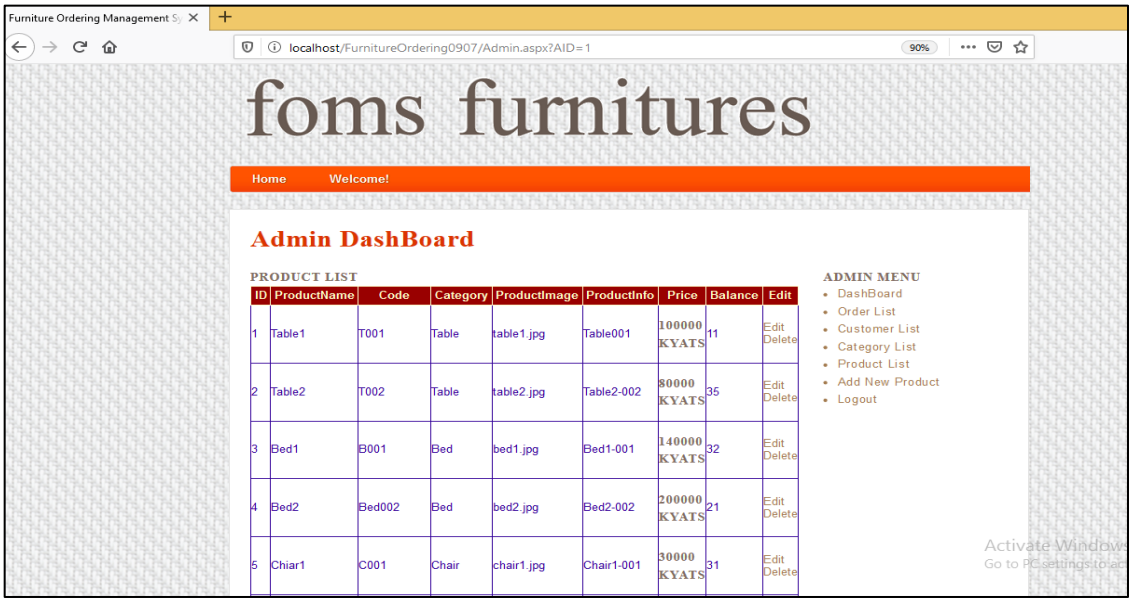

Figure 4.10: Product List (Admin Site)

4.3.2. Furniture Ordering Process at Customer Site

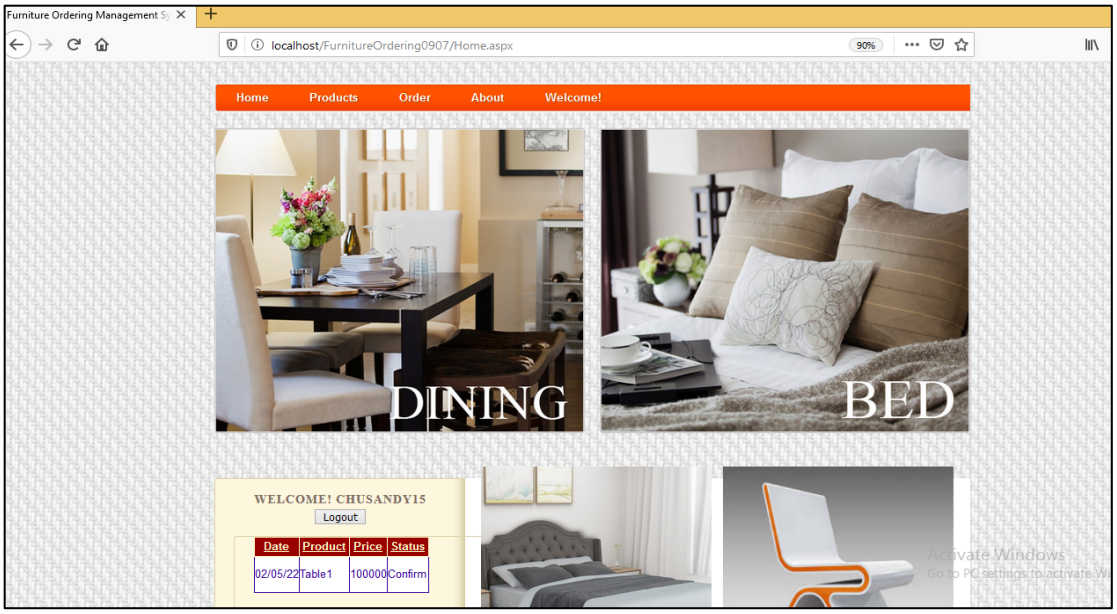

Figure 4.11: Home Page (Customer Site)

The above figure is the home page of the customer side of “Low Latency Fault Tolerance Furniture Ordering System”. The “Home page” contains “Products” Menu, “Order” Menu, and “About” Menu. Details of each menu will be explained as the following. Products Page has two frames (Left frame and Right Frame).

Figure 4.12: Product List to Order (Customer Site)

Left frame shows the category list to provide the quick search for the customer needed products. And the right frame shows the detailed product list of the user selected category. The product list shows the product information such as product code, product category, Price and a button is supported to order the product as shown in Figure 4.12.

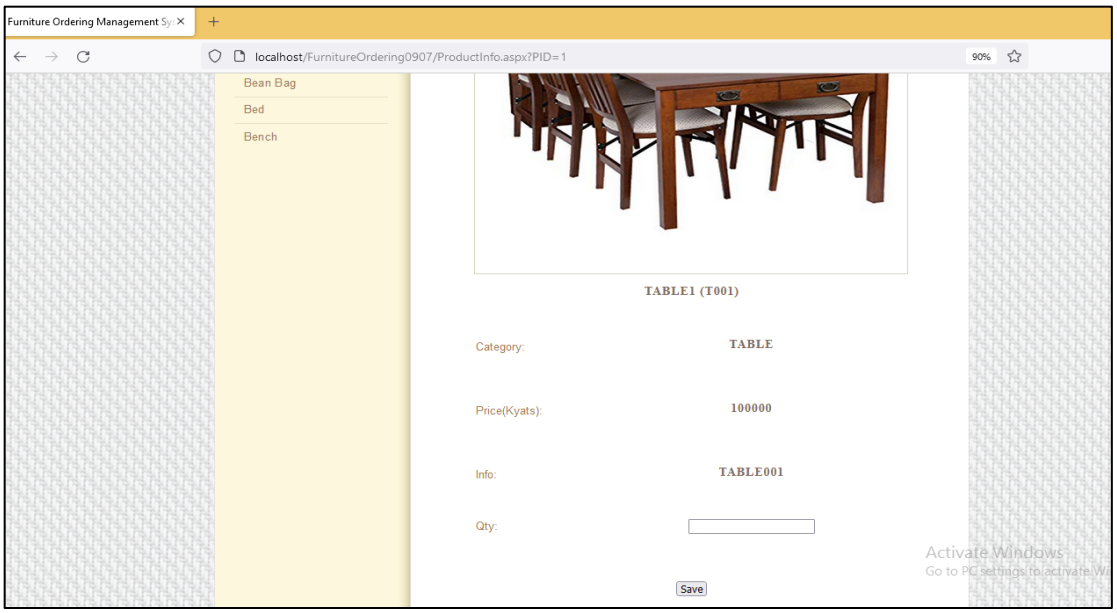

Figure 4.13: Ordering Page (Customer Site)

Figure 4.13 is the "Ordering Page" of the user selected item. In this page, user must enter the desire amount to order and "save" button to take the ordering process. The confirmed order of the user is stored in Order List table.

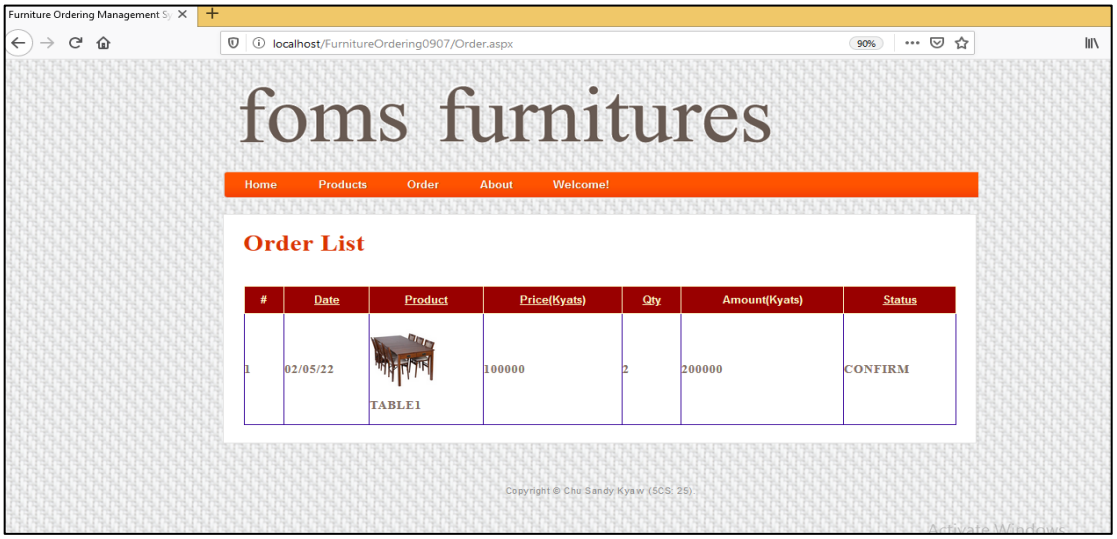

Figure 4.14: Confirmed Order List (Customer Site)

4.3.3. Confirmed Order Saving at Admin Site

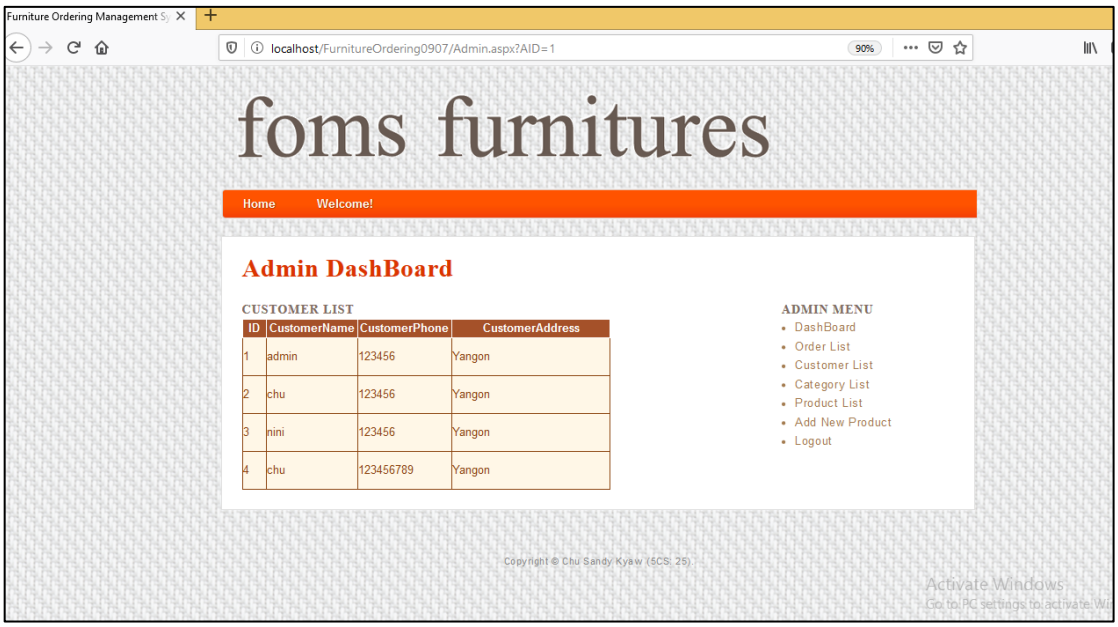

Figure 4.15: Customer List (Admin Site)

As this system is the furniture ordering system, the confirmed order customer lists are stored at the admin side for delivering process. So, this system maintains the Customer List and Order List as shown in Figure 4.15 and Figure 4.16.

Figure 4.16: Order List (Admin Site)

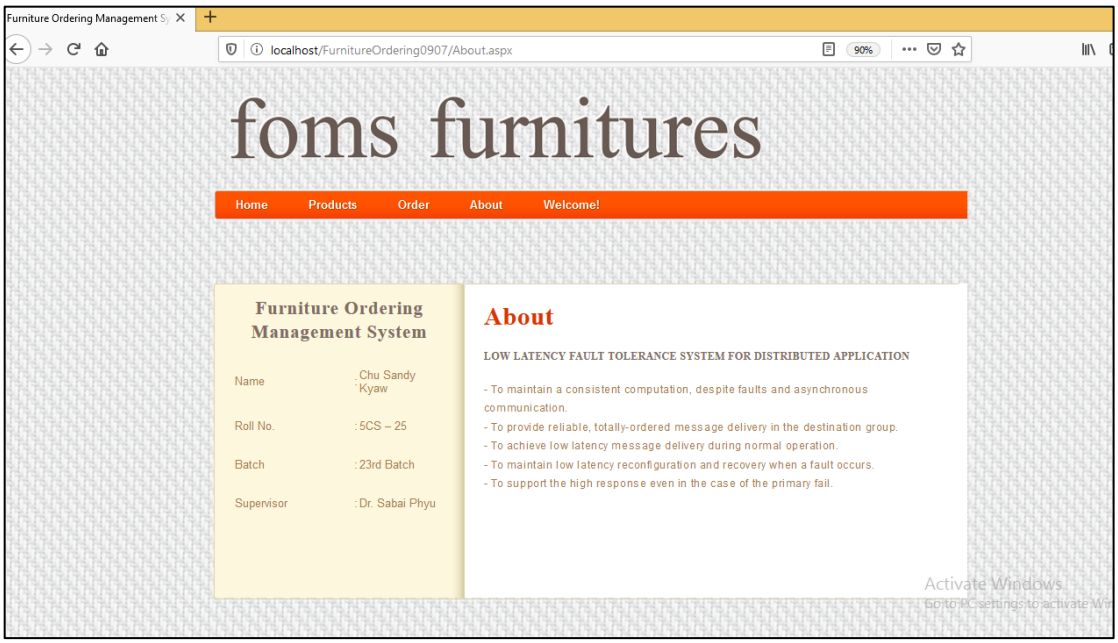

Figure 4.17: About Page of the System

The Figure 4.17 is about the page of the system. This page describes the main objectives of the proposed low latency fault tolerance furniture ordering system.

CHAPTER 5
CONCLUSION AND FURTHER EXTENSIONS

The proposed LLFT framework is underlined on the Furniture Ordering Management System. The Low Latency Fault Tolerance (LLFT) framework gives adaptation to internal failure to circulated applications sent over a wide-region organization. This framework can reproduce areas of strength for with consistency utilizing LLFT, with practically no modifications to the applications. LLFT accomplishes low dormancy message conveyance under typical circumstances and low idleness reconfiguration and recuperation when a shortcoming happens. The application straightforwardness and low dormancy of LLFT make it proper for a wide assortment of conveyed applications, especially for idleness touchy applications.

5.1. Advantages of the System

By the adequate replication presumption (i.e., each gathering contains an adequate number of copies with the end goal that in every essential view there exists something like one imitation that does not become defective), in the event that the essential becomes flawed in a view Vi, there exists a copy R in Vi that can expect the job of the essential in view Vi+1. The evidence follows by enlistment.

5.2. Limitations and Further Extensions

The proposed framework is coordinated by the client site and server site. The request exchange from the client site is conveyed to the server and the handling take set. Then, the handling result is combined as a solitary outcome and return to the client page. Thus, the beginning of the request exchange is the client site level and handling happens in server level. During handling stage, the LLFT controls the primary server or backup server consistency regardless of whether primary is connected or not. Be that as it may, the framework just underlined the server and reinforcement framework. Thus, this framework can be improved to the center framework by utilizing cloud environments.

Future work incorporates sterilization of different wellsprings of non-determinism, (for example, working framework signals and interferes) and execution improvement. It additionally

incorporates the advancement of additional mind-boggling applications for LLFT (specifically, record frameworks and information base frameworks), and the improvement of replication the board devices.

AUTHOR'S PUBLICATIONS

[1] Chu Sandy Kyaw, Dr. Sabai Phyu, "Low Latency Fault Tolerance System for Distributed Applications", Parallel and Soft Computing Journal (PSC), UCSY, Yangon, Myanmar, March 2020.

REFERENCES

[1] Alireza Souri1, Saeid Pashazadeh*2 and Ahmad Habibizad Navin3, Consistency of Data Replication Protocols in Database Systems. International Journal on Information Theory (IJIT),Vol. 3, No.4 October 2014

[2] C. Marchetti. "Software replication in three- tiers architectures: is it a real challange?", Proceedings Eighth IEEE Workshop on Future Trends of Distributed Computing Systems FTDCS 2001 FTDCS-01, 2001

[3] Database System Journal ISSN 2069-3230
Date of access: December 2020
http://www.dbjournal.ro/index.html

[4] Error_502, What is Replication in Distributed System?
Date of access: Jan 2021
https://www.geeksforgeeks.org/what-is-replication-in-distributed-system/

[5] Hua Chai, EngageScholarShip
Date of access: Mar 2021
https://engagedscholarship.csuohio.edu/do/search/?q=Database%20Replication&start=0&context=2292565&facet=

[6] Khin Kaung San, Khaing, Implementation of Database Consistency by Active Replication. University of Computer Studies, Yangon.

[7] Marius Cristan MAXILU, Database Systems Journ vol. I, no.2/2010

[8] M. Wiesmann, F. Pdeone, Understanding Replication in Databases and Distributed Systems, A.Schiper Swiss Federal Institute of Technology (EPFL) Operation Systems Laboratory IN-F Ecublens, CH1015 Lausanne.

[9] Phyoe Su Su Win, Thet Su Mon, Data Consistency of Distributed Transaction for Order Management. University of Computer Studies, Taung-Ngu.

[10] Peter von Oven. "Chapter 4 Getting Started with the Management Console", Springer.

[11] Sujoy Paul. "Pro SQL Server 2008 Replication", Springer Science and Business Media LLC, 2009.