MEETING ROOM MANAGEMENT SYSTEM ON PEER-TO-PEER OVERLAY NETWORK BY SCRIBE TO EVENT-BASED ROUTING

SU YADANAR THAN HTIKE

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

MEETING ROOM MANAGEMENT SYSTEM ON PEER-TO-PEER OVERLAY NETWORK BY SCRIBE TO EVENT-BASED ROUTING

BY

SU YADANAR THAN HTIKE

B.C.Sc.

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 supporting of many people.

First and foremost, I would like to express my deepest gratitude and many 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 thank course coordinators, **Dr. Si Si Mar Win** and **Dr. Tin Zar Taw**, Professors of Faculty of Computer Science, the University of Computer Studies, Yangon, for their support, guidance, supervision, patience and encouragement during the period of study towards completion of this thesis.

My thanks and regards go to my supervisor, **Dr. Yu Yu Than,** Professor of Faculty of Information Science Department, 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 Mya Hnin Mon**, 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 to my parents, all of 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 not been submitted for a higher degree to any other University or Institution.

---------- ----------

Date Su Yadanar Than Htike

ABSTRACT

With the adventure of the Internet, it became possible to build large-scale distributed applications due to the existence of a global, packet-based communication infrastructure. This increases in application scale by several orders of magnitude results in systems with variant nodes that require to connect and cooperate so as to realize a typical goal. This system presents an efficient method for building large-scale distributed systems (Case Study: meeting room management system). The propose method employs two underlying techniques: event-based communication and peer-to-peer overlays. Overlay broker networks are a vital a part of an event-based middleware. In this thesis, an event-based middleware architecture that uses a peer-to-peer routing substrate, as compared with a regular publish/subscribe system that incorporates a simple, scalable, self-organizing and decentralized overlay topology for distributed applications.

TABLE OF CONTENTS

		Page
ACKNOWLEDGEMENTS		i
STATEMENT OF ORIGINALITY		ii
ABSTRACT		iii
TABLE OF CONTENTS		iv
LIST OF FIGURES		vii
LIST OF TABLES		viii
LIST OF ABBREVIATIONS		ix
CHAPTER 1	**INTRODUCTION**	1
	1.1 Objectives of the Thesis	2
	1.2 Related Works	3
	1.3 Organization of the Thesis	4
CHAPTER 2	**BACKGROUND THEORY**	5
	2.1 Event-Based Communication	5
	2.1.1 Creating Topic	6
	2.1.2 Subscribing	6
	2.1.3 Publishing an Event	7
	2.2 Peer-to-Peer Overlay Networks	6
	2.3 The Concept of Replication	8
	2.4 Database Replication	8
	2.5 Selection of the Database Replication	9
	2.6 Using the Database Replication	9
	2.7 Database Replication Procedures	10
	2.7.1 Snapshot Replication	10

2.7.2 Merging Replication 11
2.7.3 Transactional Replication 12
2.8 Distributed Transaction Management 14
CHAPTER 3 CONCURRENCY CONTROL OF REPLICATION MIDDLEWARE ARCHITECTURE 16
3.1 Database Replication Middleware 16
3.2 Multi-master Replication 17
3.3 Data Partitioning 17
3.4 Middleware-level Challenges 17
3.4.1 Intercepting Queries 18
3.4.2 Statement Vs Transaction Replication 18
3.5 Consistency Problems 19
3.5.1 Lost or Buried Updates 19
3.5.2 Inconsistent Analysis (Non repeatable Read) 19
3.5.3 Uncommitted Dependency (Dirty Read) 19
3.5.4 Phantom Reads 19
3.6 Consistency Modes 20
3.7 A Local Consistency Manager 20
3.8 COPLA Programmer Library 20
3.9 Actions Performed by the Algorithm 21
CHAPTER 4 SYSTEM DESIGN AND IMPLEMENTATION OF THE SYSTEM 22
4.1 Overview Design of the System 25
4.2 The Algorithm of the Proposed System 26
4.3 Implementation of the System 27

	4.4	The Login Page of the Proposed System	28
	4.5	Main Page of the Proposed System	29
	4.6	Booking Reservation	30
	4.7	The Detail Process of Booking Reservation	31
	4.8	Change Booking Process	33
	4.9	Concurrency Controlling	34
CHAPTER 5		**CONCLUSION**	35
	5.1	Advantages of the System	35
	5.2	Limitations of the System	35
	5.3	Further Extensions	36
AUTHOR'S PUBLICATION			37
REFERENCES			38

LIST OF FIGURES

Figure		Page
Figure 2.1	Snapshot Replication	11
Figure 2.2	Working in Transactional Replication	13
Figure 2.3	Flat Distributed Transaction	14
Figure 2.4	Nested Distributed Transaction	15
Figure 3.1	Database Scale-out Scenario	16
Figure 3.2	Database Partitioning for Increased Write Performance	17
Figure 4.1	System Flow Diagram of the Propose System	25
Figure 4.2	Login Page of the System	28
Figure 4.3	Login Page of the System with Invalid Login Name and Password	28
Figure 4.4	Main Page of the System	29
Figure 4.5	Booking Enquiry	30
Figure 4.6	Reservation Form	30
Figure 4.7	Data Update Propagation by Site 1 to Site 2	31
Figure 4.8	Data Update Propagation by Site 2 to Site 3	31
Figure 4.9	Notification for Booking Success	32
Figure 4.10	Restriction to avoid double-booking	33
Figure 4.11	Booking Cancellation	33
Figure 4.12	Notification for Concurrent Accessing on same event	34

LIST OF TABLES

Table		Page
Table 4.1	Types of Hotels and Meeting Rooms	22
Table 4.2	Concurrency Control in the Same Data Item	23

LIST OF ABBREVIATIONS

PC	Personal Computer
P2P	Peer-to-Peer
ACID	Atomicity, Consistency, Isolation, Durability
MSDTC	Microsoft Distributed Transaction Coordinator
MySQL	My Structured Query Language
IBM	International Business Machines
DB2	Database 2
TDM	Transaction Data Management
RDBMS	Relational Database Management System
RAID	Redundant array of independent disks
JDBC	Java Database Connectivity
ODBC	Open Database Connectivity
COPLA	Common Object Programmer Library Access
RDBMS	Relational Database Management System

CHAPTER 1

INTRODUCTION

The large boom in computing electricity and also the emergence of high-velocity PC networks has made allocated large-scale systems viable and computing allocated today is widely deployed while building huge software program systems. Such systems, usually some variations of the standard request / reply, use distinctive interaction models. The most project with this model of interaction is its synchronous existence, resulting in very static architectures.

Any other inherent request / reply problem is its point-to-point communication version, which creates close coupling between speaking parties. A new methodology is crucial because it creates large-scale decentralized networks that should proportion to the massive internet world of today. A model well known a completely communicate or put up / subscribe paradigm supported an occasion offers a good option to do this and is becoming more and more crucial. The straightforward principle is that the system is formed from publishers who send event, and subscribers who participate in successful occasions [4].

This model helps with many-to-many verbal exchanges and high decoupling between the various events. Fully daily interaction could be a powerful version. It also contains the apex of an activity or message agent overlay network. Message agents form distribution trees that can be used to route the event to the subscribers from the publisher. Such agents depend upon the routing of the utility degree because the multicast network degree has not been commonly used now.

This system aims to develop the meeting room system by using event-based communication. Meeting room management systems incorporate software for room scheduling and interfaces. The software allows company employees or guests to book meetings online or through an application.

Software-based room booking systems could also be employed in hospitality, in studios or spas, in hotels, or in enterprises for employee booking. Another popular usage is shared workspaces, for reserving rooms or simply desk space.

Meeting Room Booking Systems feature are the following capabilities:

i. Book conference rooms electronically
ii. Set reservations or book impromptu meetings
iii. Seating assignments, or hot desks
iv. Visitor management (e.g. visitor details, sign-in, reminders)
v. Repository for room information (e.g. amenities, photos, etc.)
vi. Floor plans and maps for guests or facilities
vii. Check-in using app
viii. Real-time availability detection, up-to-date floorplans
ix. Search for available meeting rooms with filters (e.g. location, time available)
x. Usage analytics (e.g. utilization, busiest times, etc.)
xi. Booking available on the online, mobile app, or a tablet
xii. Room issue reporting (e.g. equipment failure) through the app
xiii. Integrate with digital signage or room displays / interfaces
xiv. Integrate with commonly used calendar or productivity apps

1.1 Objectives of the Thesis

The main objectives of this thesis are:

- To present and describe the event based on middleware service.
- To implement the publisher/subscriber communication between components providing a scalable and efficient event service.
- To understand how consistency is very important in database systems with multiple users.
- To implement a system in which all objects remain during a consistent state once users are accessed by multiple transactions.

1.2 Related Works

The related works of concurrency controls are discussed during this session.

In [1], processes at each exit and entrance maintain the quantity of vacancies within the car parking. Firstly, the method exams the type of event (getting into or leaving) requested for updating the big variety of vacancies. If the type of process is a "getting into event", the system will test any vacancies within the auto parking. If there is not any emptiness, the system will display the "unavailable message" to the customer. Otherwise, the system will multicast requests to any or all different tactics and wait for their replies. After the system user have all reply, the method decreases the range of vacancies by 1 and lets the car input. Then, it is multicast the up-to-date facts to all or any different hosts. And it replies all requests are saved by inside the local queue. Sooner or later, the method leaves the essential phase. If the type of process is "leaving manner", the method makes requests to all or any different processes and waits for replies. Upon getting all replies, the method increases the quantity of vacancies by 1 and lets the car depart. Then, it's multicasts the up-to-date information to all or any different approaches. And it replies all requests saved within the neighborhood queue. In the end, the system leaves the crucial phase.

Event oriented model was introduced in distributed system since 1970s [2]. Many software and operating system use graphical consumer interface based totally infrequently. Dealing room system changed into designed based totally on the difficulty of event and notification. This process affords the current price of stock on market. Whilst the adjustments on price occur, it could tell to the consumer right away by sending notification rubdown. As it turned into process based on the event and notification paradigm, the consumer does not need to couple with the gadget and therefore the conversation between the device and users is asynchronous.

Internet Indirection Infrastructure (I3) is another rendezvous-based communication abstraction [9]. It offers an application-level multicast architecture which will be seen as a kind of topic-based publish-subscribe system. Each packet is related to an identifier which is employed by sources (publishers) to send the packet, and by receivers (subscribers) to specific their interest in certain packets. This effectively decouples technique is sending from the receiving, that is, provides indirection.

Subscribing in i2 is finished by inserting a trigger to the system that tells who's interested and within which topic (id). Each id has its own rendezvous node within the network where the triggers are stored.

1.3 Organization of the Thesis

The thesis is organized as five chapters. These are as follows:

In Chapter 1, introduction of the system, objectives of the thesis, related works and thesis organization are described. **Chapter 2** presents the background theory of transaction processing. **Chapter 3** discusses concurrency control and recovery in a replication middleware architecture. **Chapter 4** expresses the design and implementation of the proposed system. Finally, **Chapter 5** presents the conclusions, advantages of system and limitations and further extensions of the system.

CHAPTER 2

BACKGROUND THEORY

The huge increase in computing power and the emergence of high-speed computer networks have made large-scale distributed systems feasible and today distributed computing is widely deployed when building large software systems. These systems use different communication models, typically some variation of the traditional request/reply paradigm. The main challenges for this form of communication is its synchronous nature which leads to very stable architecture. One of the congenital issues with request/reply is its point-to-point communication model which creates tight coupling between communicating parties [10, 11]. A different approach is needed to build services that will reach a wide range of Internet environments today. The so-called "event-based communication" or “publish/subscribe paradigm” is becoming more and more important, by giving it a better choice to do so. The basic principle is that the system consists of publishers that publish events, and subscribers that subscribe to certain events. Subscription can simply be topic-based, but more advanced schemes, such as content-based and type-based schemes, have been proposed. Events are delivered asynchronously to the receivers. This model supports many-to-many communication and high decoupling between the different parties. [9]

2.1 Event-Based Communication

Numerous forms of verbal exchange models frequently lack good transparency in time, space and synchronization, which un-appropriate for big-scale disbursed computing. Occasion- based totally communique offers an efficient approach to clear up those problems. A perfect introduction to periodic, mainly focused verbal exchange, often referred to as the model of submission / subscription. The basic function of subscribers is to remind the computer of their participation in positive activities. This is usually done by subscribe () activity on a few forms of event notification service. This form of service can be seen by publishers and subscribers as an impartial mediator. This carrier is also commonly referred to as a

message or event broker, and an ordinary pub / sub network consists of a variety of them forming an overlay group over a community layer of the body.

When a publisher wants to publish an event, it calls the event service to notify () operation, which is then responsible for routing the event through the event broker network to the interested subscribers. The concept of event service provides three-dimensional decoupling between the event publisher and subscribers: space, time and synchronization [9].

Space Decoupling (SD): A decoupled architecture allows each component to perform its tasks independently of the others, while also enabling structural variations between source and target.

Time Decoupling (TD): results in independence of actions in time so that the communicating parties do not need to be active at the same time.

When sending or receiving events, publishers and subscribers are not blocked. These properties of decoupling result in a very transparent communication infrastructure that scales well and supports a large number of subscribers and publishers.

2.1.1 Creating Topic

Every subject has a correct topic Id. The scribe node with node Id that is numerically closest to the topic Id acts as a meeting point for the related topic. When creating an issue, a scribe node asks pastry to path the CREATE message with the subject Id to the nearest number node, the rendezvous node. The appointment reviews the credentials and adds the subject to its list of topics. The topic Id is actually a hash of the subject's request to use a collision-free hash function in conjunction with the name of its author. It means that on the underlying network, the ids are dispensed equally [6].

2.1.2 Subscribing

If a Scribe node decides to subscribe to a subject, the SUBSCRIBE message will be sent using the route method where the topic Id is used as the password. This brings the message to the rendezvous of the subject. Every node along the route investigates whether the topic is already widely subscribed. If the subscription is for an unknown issue, the subscription will be sent forward to the rendezvous as an entry for it is far produced to a children's desk to consider the subscription [11].

This mechanism of subscription is very scalable to a large number of topics and subscribers per topic. In addition, many of the nodes within the network are adequately dispensed with the forwarding load, and the rendezvous factors in particular are not overloaded. Local houses are lowering the traffic created by the network [8].

2.1.3 Publishing an Event

If a publisher wants to publish an event, it calls on Pastry to use the topic Id as the key to route the event to the rendezvous point. In the peer-to-peer overlay network, the rendezvous performs access control and forward the event to the multicast.

2.2 Peer-to-Peer Overlay Networks

Peer-to-peer overlay networks have emerged as an efficient communique approach. The peer-to-peer (P2P) networks offer numerous exciting aspects like fault-tolerance and absolutely decentralized manage. This allows them to adapt nicely to exceedingly dynamic networks, i.e., networks, wherein nodes may additionally be a part of and leave at will each time, and they scale well to nowadays internet-extensive surroundings.

An abstract P2P overlay network architecture as below;

i. **The Network Communications layer** describes the network layer is the ad-hoc manner that connects the network features of desktop devices connected to the Internet or small wireless or sensor-based devices. The dynamic nature of peers poses challenges in the telecommunications paradigm.

ii. **The Overlay Nodes Management layer** covers the peer management, including discovery of peers and routing algorithms for optimization.

iii. **The Features Management layer** deals with the security, reliability, fault resiliency and the areas of consolidated resource availability aspects of maintaining the robustness of P2P systems.

iv. **The Services Specific layer** supports the back-end P2P infrastructure to perform computational tasks, scheduling of parallel, content and file management and the application specific components. Meta-data describes the content stored and the location information across the P2P peers.

v. **The Application-level layer** relates the tools, applications and services that are implemented with specific functionalities on top of the underlying P2P overlay infrastructure.

2.3 The Concept of Replication

It starts with the word "Replication", which represents the process of software or hardware components to enhance accessibility. If the same data is copied, the calculation is duplicated. Access to a duplicate organization means access to a non-duplicate organization. Replication is transparent to an external user. In addition, the failure of replicates is kept as hidden as possible. Systems that duplicate data are active or passive. When each request is made on a duplicate, it is transferred to another duplicate. Learn about the master-slave scheme to handle all requests. On the other hand, if a duplicate executes a request and distributes a new status, it is a multi-primary scheme (called a multi-master in the database field). [2]

The Data Replication process uses the same pieces of data; Replicates should not be confused with the backup process as they are frequently updated and quickly lost. Copying, on the other hand, saves a large amount of unchanged data copy.

2.4 Database Replication

Database replication is the process of transforming a database in different places without having to copy the entire database. A database server maintains a master copy of the database, and database servers maintain a slave copy of the database. Two or more copies of a database exist at the same time. [3]

The original database is called the Design Master, and each copy of each database is called a replicate. There is only one Design Master in a replica set. Synchronization contains data in every copy of the database. When duplicates are duplicated, only updated data is updated. Design and make changes to objects in Design Master. [1]

Database entries are sent to the master database server and duplicated on the slave database servers. Reading databases offers great performance. In addition, if a master database server is not available, slave database servers can transfer master roles, which can improve database replication. [3]

2.5 Selection the Database Replication

Replication is the implementation and maintenance of copying can be very confusing with a lot of database servers. Implementing replication can also be complicated by the application architecture. However, replication can be put to good use. Database backup is well suited for the required business solutions. Share data between remote offices. A corporate database copy sends to each satellite office over a wide area network (WAN). All remote replicas are identical to those at corporate headquarters. Individual replicas can retain information that is not included in other replicas. Share data between different users.

Users can synchronize new information entered in the database while out of the office by setting up a corporate network and an electronic link at any time. Users can call the network, replicate and work in the current version of the database. Because it only sends incremental changes during replication. The latest information will save the time and money. By using partial replicas, can be done simultaneously.

i. Make the Server Data more Usable:

Immediate updates need the data, replication can be used on the main server to reduce network load. Introducing a second server with a personal copy of the database improves response time. The changing requirements adjust the replication. Copying requires less centralized management.

ii. Distribute Solution Updates:

When copying It automatically replicates not only the data in the tables but also the objects. If the design of the database changes, next time, send changes it. The full version of the software distributes do not need it.

iii. Backup Data:

First of all, duplicating a database seems very similar to copying a database. However Initially making a complete copy of the database, later it is repeated in the normal interval between the original objects. If the original database is damaged, this copy can be used to recover it. In addition, Users of any form can access it through a backup process.

iv. Support Internet or Internet Replication:

It can configure of the Internet or Internet server to be used as a hub for distributing changes to the included replicas. [1]

2.6 Using the Database Replication

Database replication has many benefits and can solve many problems in distributed database operations. In some cases, database replication is not supported if replication is below standards:

i. Many replicates have frequent updates of existing records: Solutions that many record updates are more likely to result in discrepancies. If different users make changes to the same log, it is certain that discrepancies will occur at the same time. Conflicts can be applied over time.

ii. Data consistency is important at all times**:** Solutions that fund transfer; payment methods are used with accurate information at all times, such as tracking airline bookings and package shipments. Payments make the replicate, but there is no support for pending payments within a replica. During the replication, the exchange information is the result of the payment, not the payment itself.

2.7 Database Replication Procedures

Database replication can be performed in at least three different ways:

i. **Snapshot Replication:** Data on one database server is plainly copied to another database server, or to another database on the same server.

ii. **Merging Replication:** Data from two or more databases is combined into a single database.

iii. **Transactional Replication:** Users obtain complete initial copies of the database and then obtain periodic updates as data changes.

2.7.1 Snapshot Replication

This type of Database Replication is a system installation. One-way replication allows data to be processed in batch format over time. The registered server can run for a period of time without updating. Running without data updates for a period of time is called latency.

As one for example, a retail store uses replication to maintain a specific statistic. Because statistics can be managed on a weekly or monthly basis, retailers can run the central server for several days at a time. The subscriber can use it for a long time without updating. Cheaper than other methods. Flashback has the advantage

of not having a key in duplicate tables. A snapshot reads the database and runs in the distributor's running folder. These files are called flash files and contain some additional information that could be used to assist the registry server. [5]

Current data is essential for pricing, listing, and listing. Instant copying of data, such as online catalogs or decision data, is often used.

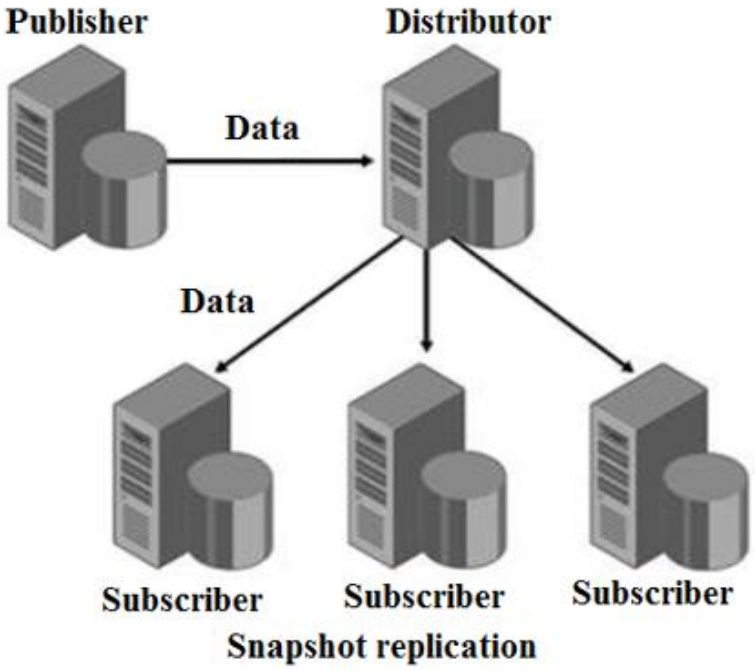

Figure 2.1 Snapshot Replication

Snapshot replication is helpful when:

- Data is generally stable and does not change frequently.
- Accept copies of data that are out of date for a period of time.
- Restoring the entire data is copying a small amount of data.

2.7.2 Merging replication

Merge replication is the process of distributing data from Publisher to Subscribers, allowing the Publisher and Subscribers to make updates while connected or disconnected, and then merging the updates between sites when they are connected.

Merge replication allows various sites to work autonomously and at a later time merge updates into a single, uniform result. The initial snapshot is applied to Subscribers, and then changes are tracked to publish data at the Publisher and at the

Subscribers. The data is synchronized between servers continuously, at a scheduled time, or on demand. Because updates are made at more than one server, the same data could be updated by the Publisher or by more than one Subscriber. Therefore, conflicts can be occurred when updates are merged.

Integrated copy includes original and custom options for conflict resolution. When a conflict occurs, a resolver is invoked by the Merge Agent and determines which data will be accepted and propagated to other sites.

Merge Replication is helpful when:

- Many registrars need to update their data at different times and distribute these changes to publishers and other subscribers.
- Subscribers must be able to receive data; Offline changes need to be synchronized between the publisher and other subscribers.
- Many sites have conflicting issues when updating data. (Because the data is filtered into fragments and uploaded to different subscribers). However, in the event of a conflict, the ACID properties are violated. [1]

2.7.3 Transactional Replication

The payment form copy, which is considered to be the opposite of instant copy, works by sending changes to the subscriber. As one for example, SQL Server runs in a database using Transact-SQL statements. Each completed statement is called a payment. In trading, each remittance is replicated to the subscriber. The backup process is done so that all the changes can be accumulated and all the changes can be sent in a timely manner. This replication can be used in environments with lower latency and higher bandwidth connections. If the server fails to connect for a copy, the transaction history can no longer be managed and a trusted connection is required. The initial copy starts with a snapshot. The copy is then updated with transfers. The snapshots can the update frequently; or the snapshot can choose not the update after the first copy. Once the initial snapshot is taken, an account reader agent is used to read and distribute the payment history of the published database. The distributor agent then transfers the payment from the publisher to the subscriber.

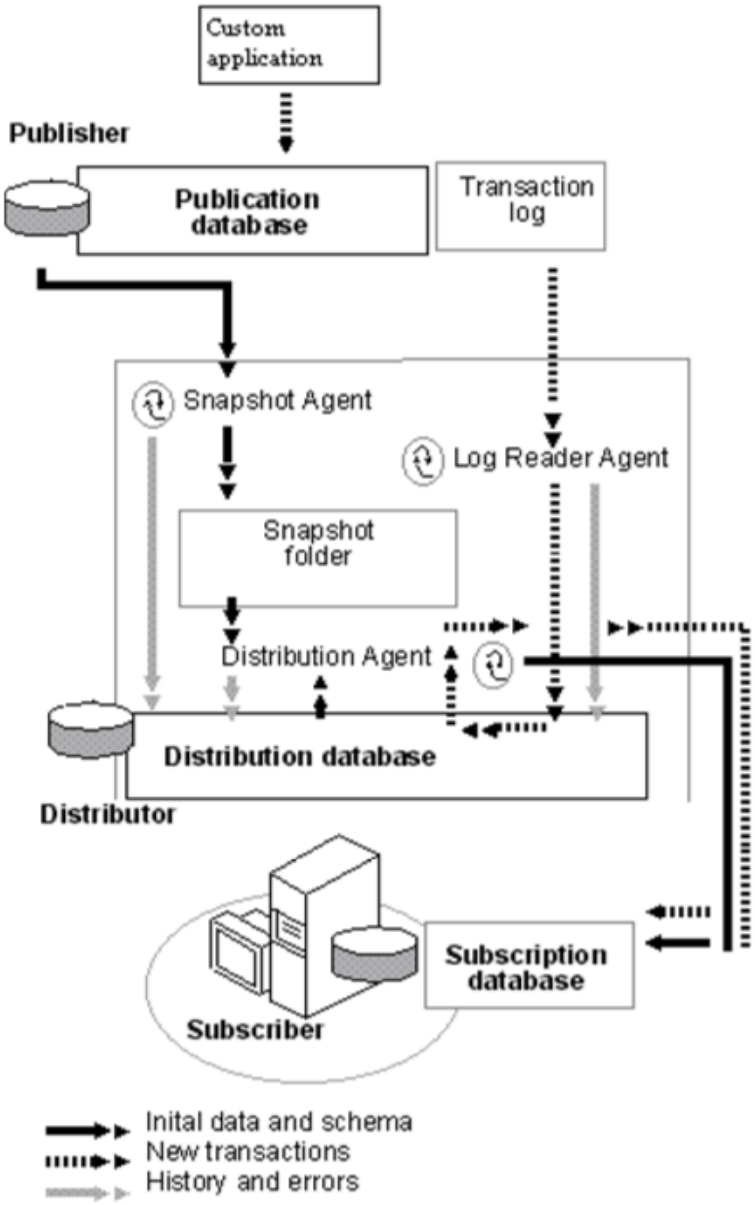

Figure 2.2 Working in Transactional Replication

Updating and transferring subscribers: The standard transfer method of replication works in the same way, but provides subscribers with data updates. When a registrant changes the internal data, SQL Server uses Microsoft Distributed Transaction Coordinator (MSDTC) to enable publishers to perform the same payments. This process may change the published data to the subscriber from time to time if required. Transferring subscribers with updated updates requires a reliable connection to medium to high bandwidth. [5] When deposit and withdrawal copying helps; Distribute enhancements to subscribers. Require ACID properties to be traded. Subscribers are often linked to trusted publishers. [6]

2.8 Distributed Transaction Management

The purpose of trading is to ensure that everything managed by a server is consistent in the event of a breakdown. The server must guarantee the results of both billing and persistent storage; Or be able to undo the consequences of a crash. It can be recovered after a server crash is called recoverable. The functionality on many different servers is distributed through a clone transaction. There are two types of distribution.

- Flat transaction
- Nested transaction [2]

In a flat transaction, the client performs on more than one server. for example, In Figure 1, Payment T is server X; A flat payment for operations on objects in Y and Z. It completes each of its requests before moving on to the next one. Therefore, each payment transaction receives a series of access to server objects. [2]

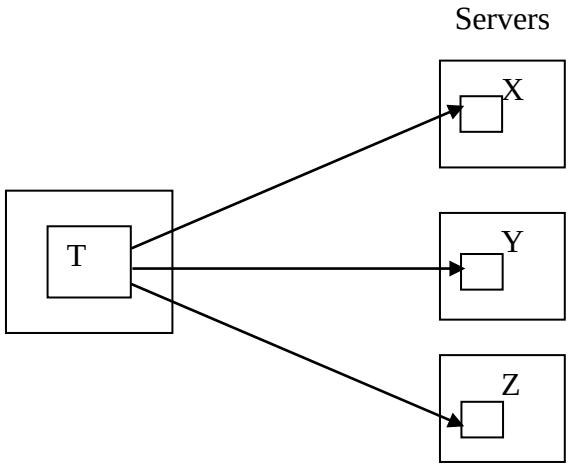

Figure 2.3 Flat Distributed Transaction

In a community Top-tier payments can open sub-branches, and each sub-branch can open additional sub-branches to the depths of the nest. [2]

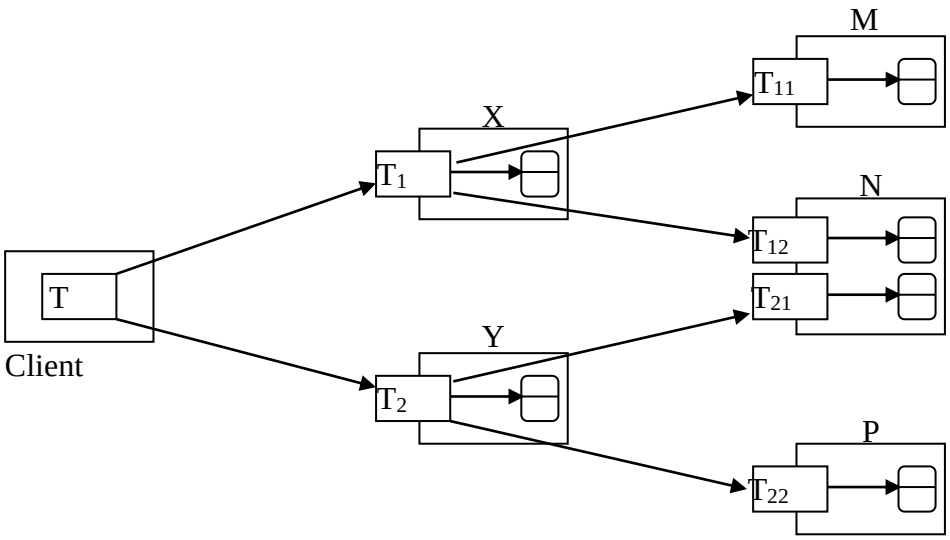

Figure 2.4 Nested Distributed Transaction

Figure 2.4 shows the payment receipt of a customer who opens two sub-items T1 and T2 on server X and Y. Transmission subdivisions T1 and T2 transfer objects to T11, T12 T21 and T22; Servers M; In the N and P. nested cases; T1 and T2 are synonymous because they can operate simultaneously on the same level of payments. The four payment sub-branches are T11; T12 T21 and T22 are performed simultaneously. [2]

CHAPTER 3
CONCURRENCY CONTROL OF REPLICATION MIDDLEWARE ARCHITECTURE

3.1 Database Replication Middleware

Database replication is often used to enhance reading or writing skills while improving both reading and writing skills simultaneous is a more challenging task.

Figure 3.1 shows the master-slave copying, a popular technology - accessing the slave node and sending updates. If the application is resilient, any data can be read at any time. The master node accepts all updates and adds slave nodes. Examples that support asynchronous master-slave copying include Microsoft SQL Server copying; Oracle Streams; Sybase Replication Server; MySQL emulation; IBM DB2 Data Propagator; Golden Gate TDM platform and Veritas Volume Replicator. Older DB systems still need readability. Replacing a DB is expensive. Satellite databases are designed for such purposes. In an e-commerce application, Although the main database is maintained, In the search for a catalog, there are few; Upload replicas. Partial replication is used. All orders must be in the main database only. User can use the database within the RDBMS, but the user must select a separate database duplicate.

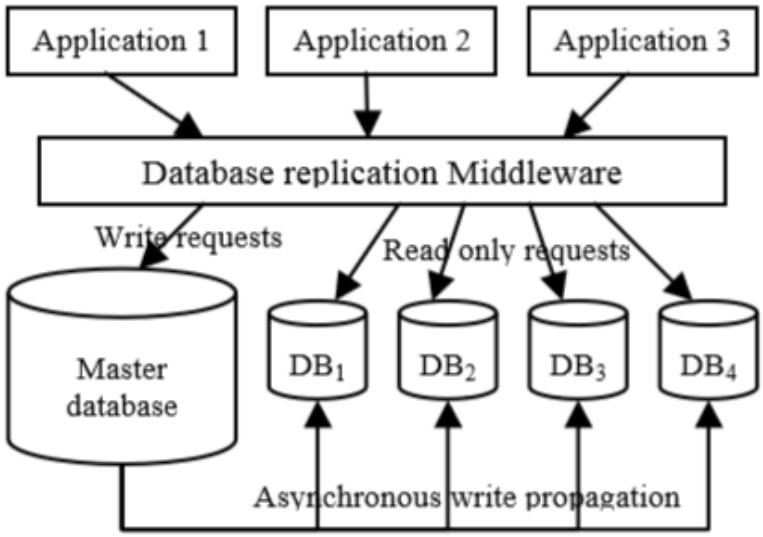

Figure 3.1. Database Scale-out Scenario

3.2 Multi-master Replication

The multi-master copy database provides both read and write capabilities. Build a centralized database that does not need to be theoretically modified. Duplicates must be able to perform payment transactions simultaneously. Only then will you be able to make updated payments. Payments can be controversial and can lead to delisting and restriction [18]. Although applications avoid different payments, Research is still underway to address this issue at the replication middleware layer. Update payment is still restricted. Not all updates have been updated yet.

3.3 Data Partitioning

Data write method is used for writing. Figure 3.2 duplicates the data into three different partitions. Common divisions are divisions; List and hash classification techniques. These benefits are similar to RAID-0 for disks, increasing read latency by competing sub-questions in each section.

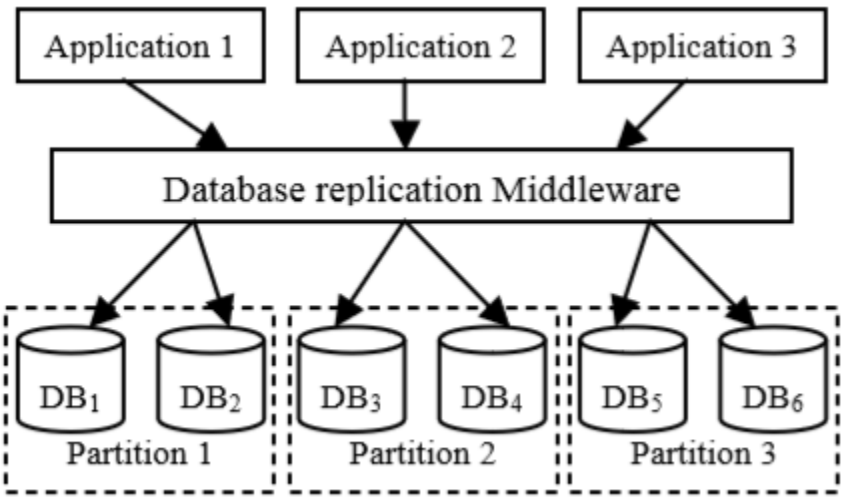

Figure 3.2. Database Partitioning for Increased Write Performance

3.4 Middleware-level Challenges

Middleware-based copying is used between application and database engines. There are several designs to intercept queries and express nodes. It describes their strengths and weaknesses.

3.4.1 Intercepting Queries

Because the database protocols change over time, the driver on the application side must change the interrupt questions. The new driver on the application side must perform breakthrough support or load balancing. The protocol version platform may vary from one language to another. for example, MySQL JDBC; Each ODBC and Perl driver has a way of interpreting its flaws and protocols.

3.4.2 Statement Vs Transaction Replication

Each time a multi-master copy is updated, a collection of payment clerks is captured and duplicated for payment. Both approaches are made with real-world applications. The undisclosed questions are: Duplication requires the same result to be updated. SQL publishing officially returns different results on different duplicates. Macros related to 'now' or 'present' provide real-time replication. Query rewrite techniques solve this problem by replacing the macro with the hard-coded value common to all duplicates. All replicas can be executed at the same time and provide results. Database code updates are usually performed using startup. The initials need to be declared each time the database layout changes. Applications can also have performance issues in addition to management. The application needs to rewrite the initials. Duplicate entries need to be avoided. Write-set retrieval does not change updates, the database configuration can change the replication. In addition, most data structures are not reversible (for example, incremental numbers in a payment transaction will not be reduced when reversed). Duplication ensures that it is updated. By copying the payment. If the application is not accurate, the values may be different. Locks and capabilities are issues with duplication. in particular, the lock is at the table level only. Capacity is limited. Smaller details need to be made into the components of the database logic in the middleware. In addition, Secondary software locking system is incompatible with the basic database lock, causing deadlocks.

3.5 Consistency Problems

Consistency problems causes by concurrent processing include

i. Lost or buried Updates
ii. Inconsistent Analysis (Non repeatable Read)
iii. Uncommitted Dependency (Dirty Read)
iv. Phantom Read

3.5.1 Lost or Buried Updates

This problem occurs when reading and updating two or more payments on the content in the shared database. Each payment was not noticed by the other payments. If the first payment reader reads it and reads the second payment for the update but does not know which payment to make first before the first payment is completed, that update will be lost.

3.5.2 Inconsistent Analysis (Non repeatable Read)

If a payment transaction reads the same data more than once, the same value should always be read. A second transaction may not be able to read it again because it has been updated multiple times for the same data. In a consistent analysis, whenever two or more identical items are exchanged for another payment, cannot read again.

3.5.3 Uncommitted Dependency (Dirty Read)

Once a payment has been updated, it is not possible to make another payment. Dirty read is like consistent analysis.

3.5.4 Phantom Reads

If a transaction does not change, it looks for data that is different from the previous one. Phantom reads add or delete trading classes.

3.6 Consistency Modes

It can be defined as a "transactions" program made on the same database connection. Each of these "transactions" can be made in one of the following unified modes:

Plain consistency - This mode does not allow write access on objects. All read accesses made in this mode are guaranteed that follow a causal order. On the other hand, this mode does not restrict on the current objects being read. As a result, it may be outdated.

Checkout consistency - This mode does not guarantee isolation, but it is similar to the traditional sequential consistency. Therefore, many sections have read a given object, One of these sections is allowed to upgrade its access mode to "Write". However, If these two sections have promoted their access modes from reading to writing, one of them will be deleted.

Transaction consistency - In this mode, the regular transaction guarantees: atomicity, sequential consistency, isolation and durability, are prescribed. A session always starts in plain mode. If the guarantees provided in this mode are not sufficient for the application, It can increase its consistency mode to checkout or transaction. In these two modes, all accesses are temporarily stored until an explicit call to the commit () or rollback () actions are made. Once one of these actions have been operated, the session automatically returns to plain mode. Therefore, the programmer can select the consistency mode of each session in which programmer writes in application, and this consistency mode may vary as needed during a session is running.

3.7 A Local Consistency Manager

Regional Unity Manager - Included in this layer Many unity protocols are allowed only one at a time. All compatible protocols share some characteristics. This is done during the compatibility check. If an operator is allowed, its updates are integrated with other Global Data node components. It depends on the functionality of the conventional protocol. This component manages all communications between the Global Data databases.

3.8 COPLA Programmer Library

The COPLA Programmer Library is the layer used by Global Data applications to access system services. Provides some capacity building kits and sections.

3.9 Actions Performed by the Algorithm

Algorithms include integrated control and retrieval; The two functions are grouped into different components: first, algorithm control, and second, Recovery process [1].

Concurrency Control

A machine executes a lock request. This function checks to see if it is compatible with the lock. In that case, the lock will need to be removed if new locks are requested. If a machine is blocked, you will not be asked to lock it. This can happen whenever the operator waits for a lock. At this point, the algorithm summarizes the waiver lock policy. The only thing with multiple sessions is requesting a read-lock. Therefore, the machine will not log in until the readable locks are generated. Special issues arise when a section is a text lock or a copy lock. The section is usually waiting to receive a reading lock or a copy lock. Blind writing is not allowed and will not wait to be locked. Due to uncertain policy. The copy lock will be in the final state.

The two reasons can be cancelling an operation: The end user decides to cancel the current host; In the first case, The task of the machine is to delete the changes in the RDBMS; Remove all locks from the machine. Unexpectedly, message exchanges with other COPLA sites must be deleted and deleted.

Recovery Actions

Related to the "ACID Principle" payment paradigm to enable recovery. A database is consistent if it includes the results of successful payments. Such a situation is called logical consistency. A payment must have a complete effect. Database update can be created a consistent.

Retrieval and cancellation transfers will be considered. The encounter situation is shown in Figure 3.1. Payment T1; T2 and T3 will survive before the crash. Recovery after system failure must actually be in the database. T4 and T5 will do it. The deal was classified as an atomic bomb. They disappear like never before. So, after the system failed, there was no choice. All incomplete payments must be removed from the database. In order to comply with these principles, a consistent database of recurring payments must be in place.

CHAPTER 4

SYSTEM DESIGN AND IMPLEMENTATION OF THE SYSTEM

The objective of this chapter present design is event publication, subscription and implementation analysis of the proposed system. The meeting room management system is created as a responsive application on a local server, which can be accessed from a remote server on the network. In this system, hotel staff can be made reservations using this application to schedule meetings. This booking reservation system keeps all meeting requests connected to a database, so Meeting Organizer can see transparent booking availability. Each booking entry has a hotel name, room type, meeting title, customer name, phone, NRC, date of book and time. In the proposed system, there is an organization which has five number of hotels and four types of meeting room as described in Table 4.1.

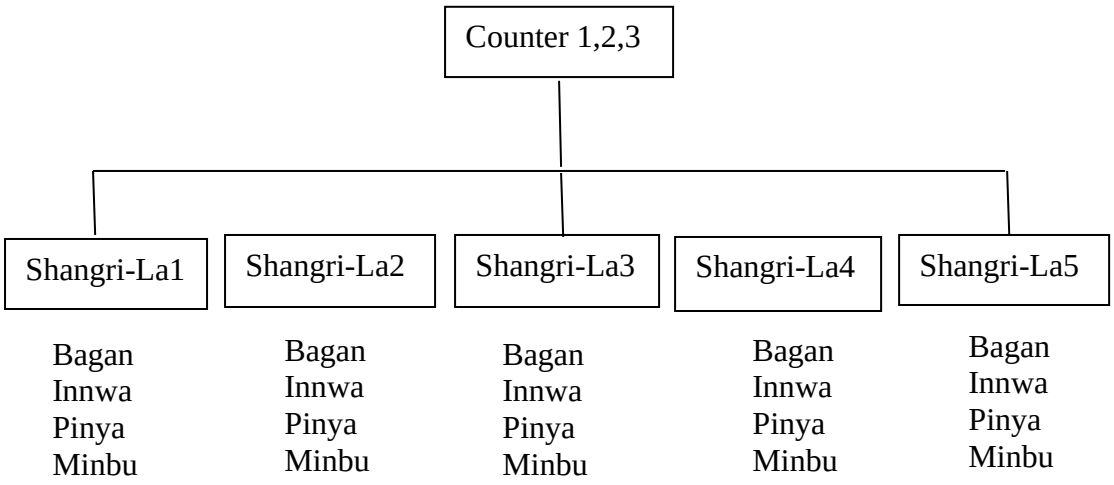

Table 4.1 Types of Hotels and Meeting Rooms

Hotel staff at the counter will be provided with meeting room scheduling and management, which will be the main focus of the system. The structure of this system allows customers to come to counter 1, 2,3 at the hotel head office and make a reservation for the meeting rooms of sub-hotels.

In Table 4.2, site1, site2, site3 are represented to user1, user2, and user3. When user1 request subscribe hotel A's meeting room at time T1 and user3 also subscribe hotel B's meeting room at time T2 and then grant subscribe at time T3 for each requested user, respectively. After accessed the subscribe, the system automatically releases the requested information of each user. There can be multiple subscribe by multiple users on the same data. After that, user 1 continue to request booking for hotel A's meeting room. The system checks for concurrency and if no concurrent occurs, the system grant for publishing and process the booking event. If concurrent occurs, the system will be solved by Time Decoupling (TD). After all finish, the system will search respective user by Space Decoupling (SD) and will send publishing result to all respective users by Event Broker.

Time	Site 1	Site 2	Site 3
T1	Subscribe(A)		
T2	-		Subscribe(B)
T3	Booking Request(A, 9:00-12:00)	Subscribe(A)	-
T4	Check for Concurrency	-	Subscribe Finish
T5	No Concurrent Booking	-	
T6	Grant for publishing	-	
T7	Process the event	-	
T8	Search respective users by SD	-	
T9	Send publishing result to all respective users by Event Broker	Updated Scribe message	
T10	Commit	Subscribe Finish	
T11		Subscribe(C)	
T12		-	Subscribe(C)
T13		Booking Request(C, 13:00 – 16:00)	-
T14		Check for Concurrency	Booking Request(C, 13:00 – 16:00)
T15	Subscribe(C)	Has concurrent publishing	Check for Concurrency
T16	-	-	Has concurrent publishing.
T17	-	Solve by TD	Solve by TD

Time	Site 1	Site 2	Site 3
T18	-	Grant for publishing	Abort by TD
T19	-	Process the event	
T20	-	Search respective users by SD	
T21	Updated Scribe message	Send publishing result to all respective users by Event Broker	
T22	Subscribe Finish	Commit	

Table 4.2 Concurrency Control for Site 1

4.1 Overview Design of the System

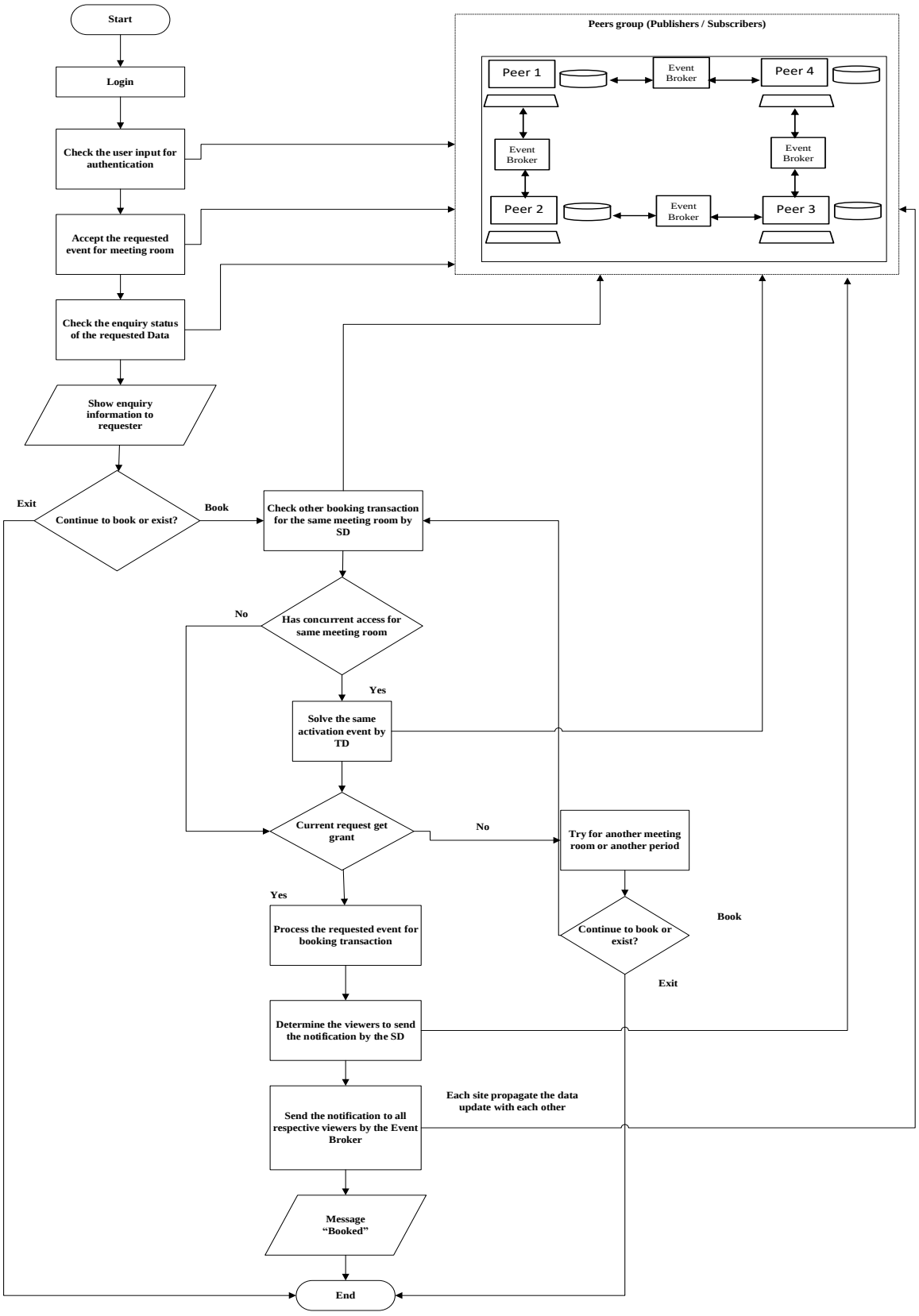

Figure 4.1 System Design of the Proposed System

4.2 The Algorithm of the Proposed System

Enquiry Function

Input Data: hotel id, room type, date time

Output: room status

Step 1 check history data table, x;

if (x.count > 0)

display unavailable message;

else

display available message;

end if

Reservation Function

Input Data: meeting title, customer name, phone, NRC

Output: booking result

Step 1 check concurrency for room;

if (concurrency is true)

solve the same activation event by Time Decoupling;

display selected room and date are held by earlier user message;

else

//room status is available;

determine the Subscribers to send the notification by the Space Decoupling;

//data replication occur

send update information to Subscribers by Event Broker network;

display booking finish message;

end if

4.3 Implementation of the System

Personnel participants want which will log in and e-book meetings, even when they're far from the workplace. Advancements in facts propagation technology have also brought about an increase in assembly room booking system utilization. Blessings of assembly room booking management device:

i. Save time
ii. Prevent errors
iii. Monitor meeting room usage

The detail processing steps of the proposed system are shown in the following Figure 4.1.

In this meeting room management system, each peer station can be accepted by the user request and checks the concurrency with the other peers' holding events. If there is no concurrent processing for the user request, the publisher of the user request holding peer publish the transaction processing event locally. After updating locally, the event broker of the peer informs the other peers' subscribers and then propagate the data update to all peers consecutively. If there is concurrent processing for user request, the system grant a chance to the user to choose the other meeting room or change the appointment time for that room or can exit from the system.

4.4 The Login Page of the Proposed System

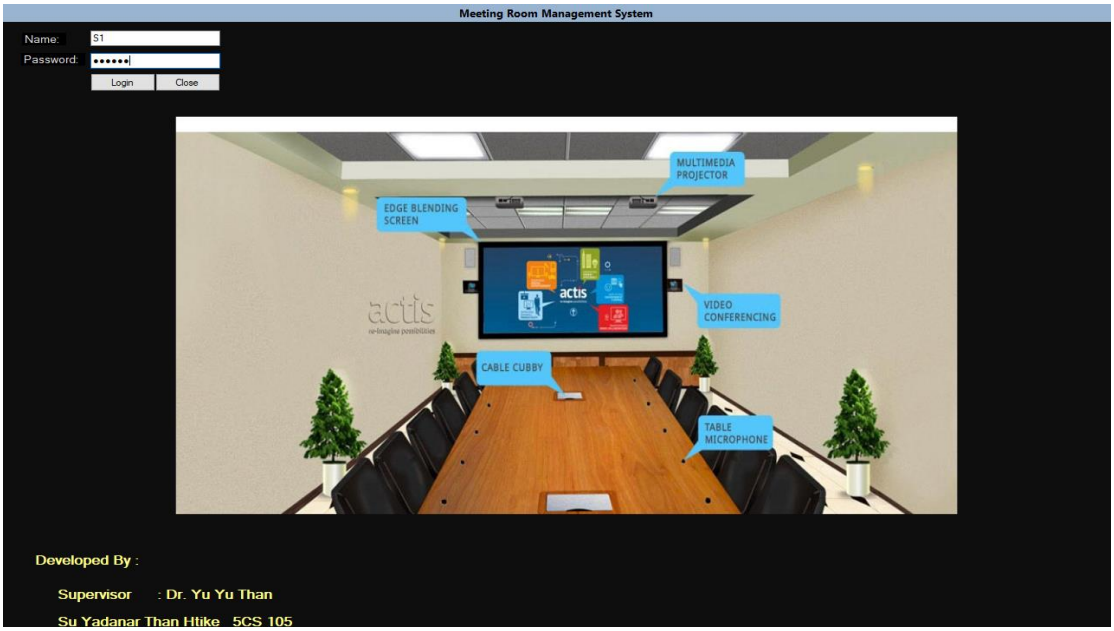

Figure 4.2 Login Page of the System

The proposed system login page is shown in Figure 4.2. The login page allows a user to gain access to an application by entering username and password. If the user's credentials do not match, this system will display a warning as shown in Figure 4.3, and after successful validation system user will see the secure part of the application as shown in the following Figure 4.4.

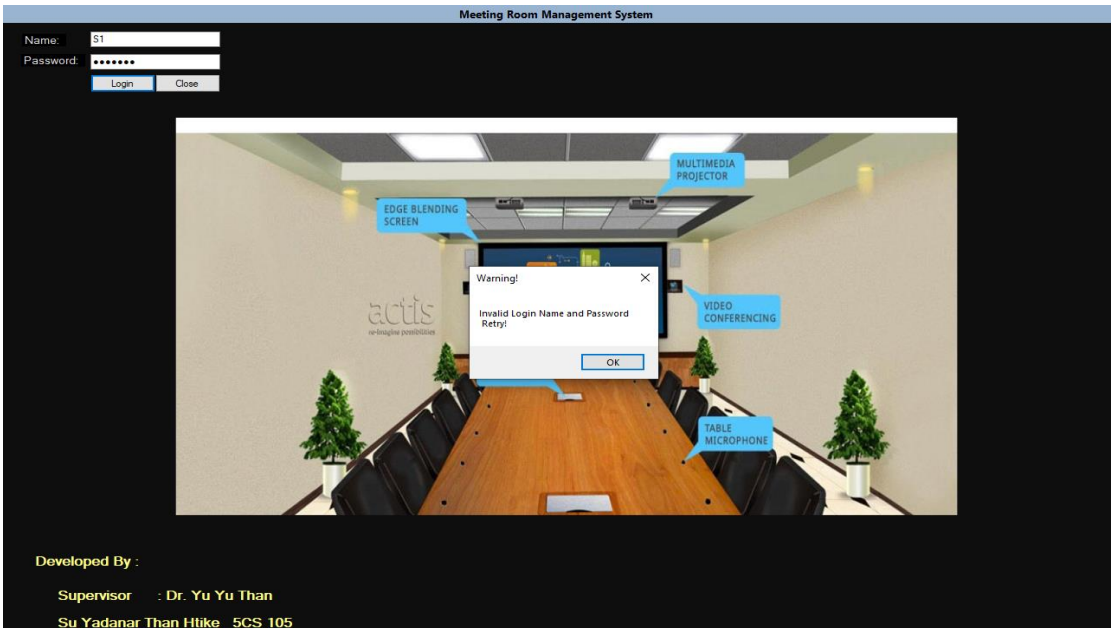

Figure 4.3 Login Page of the System with Invalid Login Name and Password

4.5 Main Page of the Proposed System

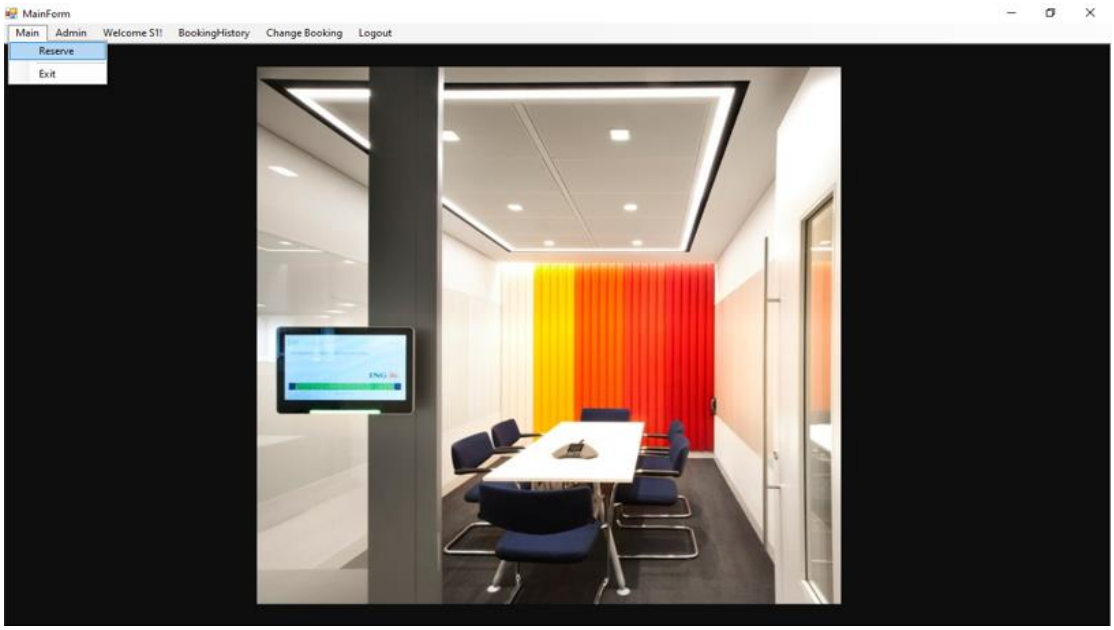

Figure 4.4 Main Page of the System

The main form in the proposed system has five menus namely, Main menu, Admin menu, Booking History menu, Change Booking menu and Logout menu respectively. Main menu makes it easier to book a room, check availability. Admin menu can be used by the admin user to setup the hotel and room type. This system can also support Change Booking menu that can be used to change the reservation, that can be received from Booking History menu allows the system user to view the detail booking history to cancel. The last menu of the system, Logout menu, to terminate a login session.

4.6 Booking Reservation

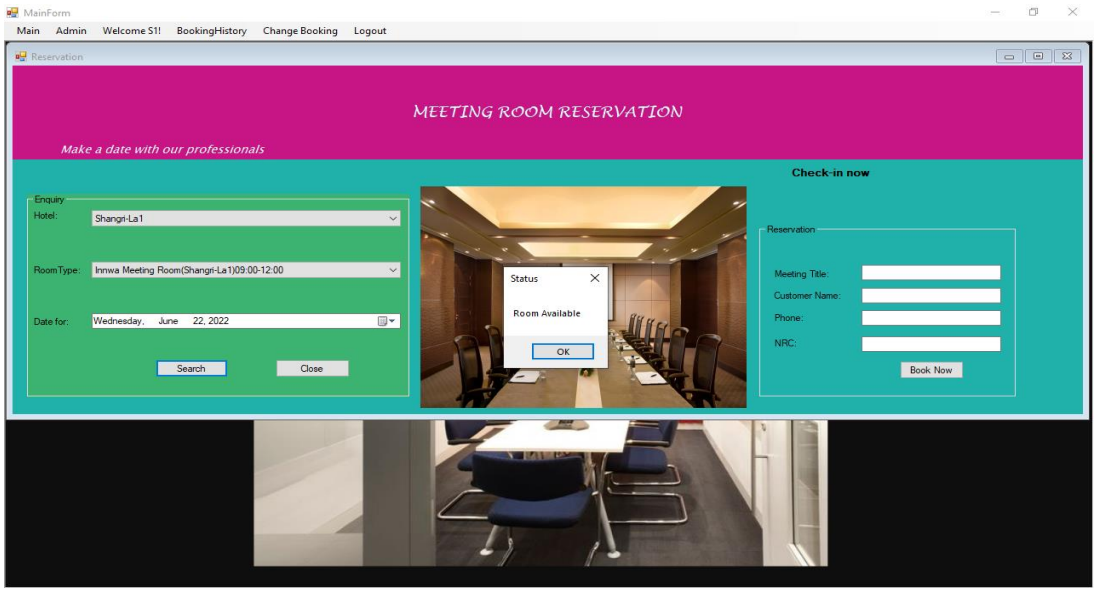

Figure 4.5 Booking Enquiry

Includes the following information when requesting a booking a meeting room: Hotel ID, Room Type and Date, Time. This System reviews the user request information and ensures the date, time, and location match the user request. The System will then accept or decline the request. If the request is not approved the Meeting Organizer is notified and the process starts over. If the request is accepted, the system sends a notification message to the Meeting Organizer stating the meeting room request was approved. And then Meeting Organizer needs to fill the reservation form as shown in Figure 4.6.

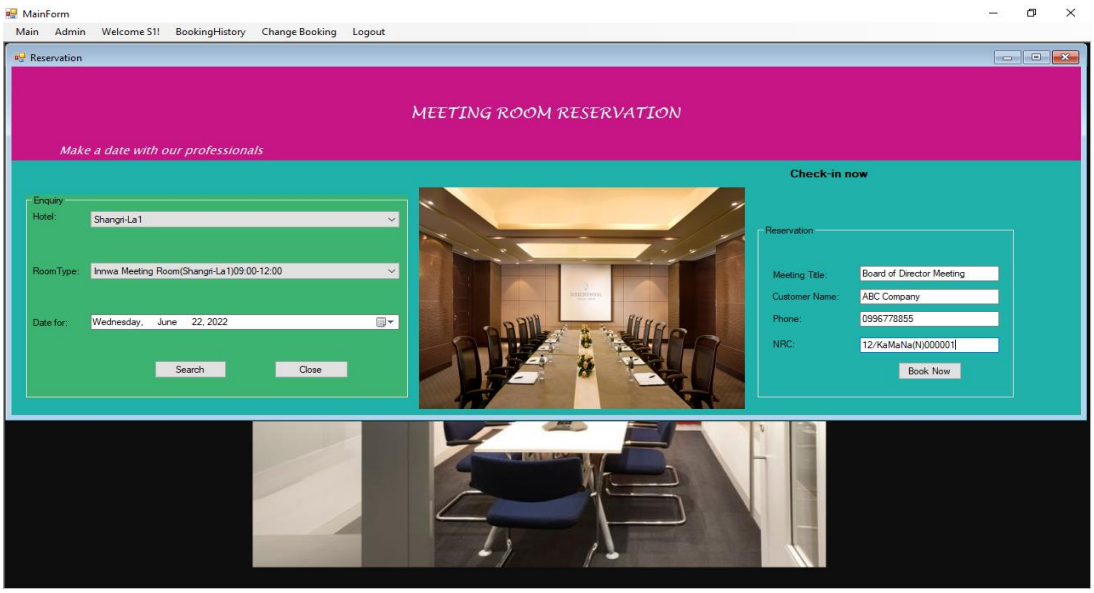

Figure 4.6 Reservation Form

4.7 The Detail Process of Booking Reservation

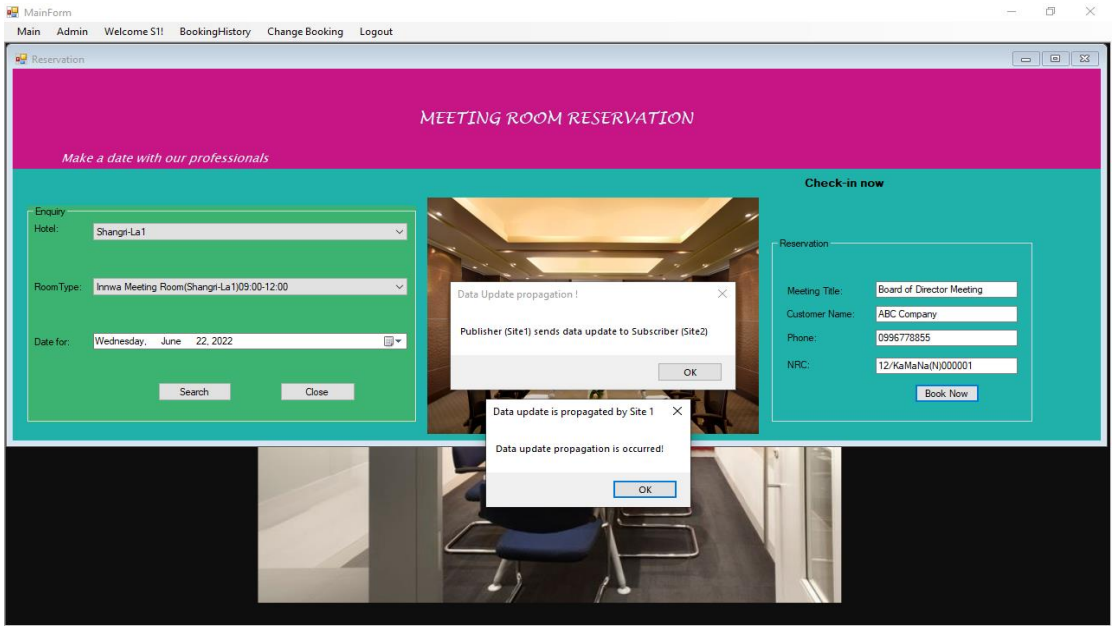

Figure 4.7 Data Update Propagation by Site 1 to Site 2

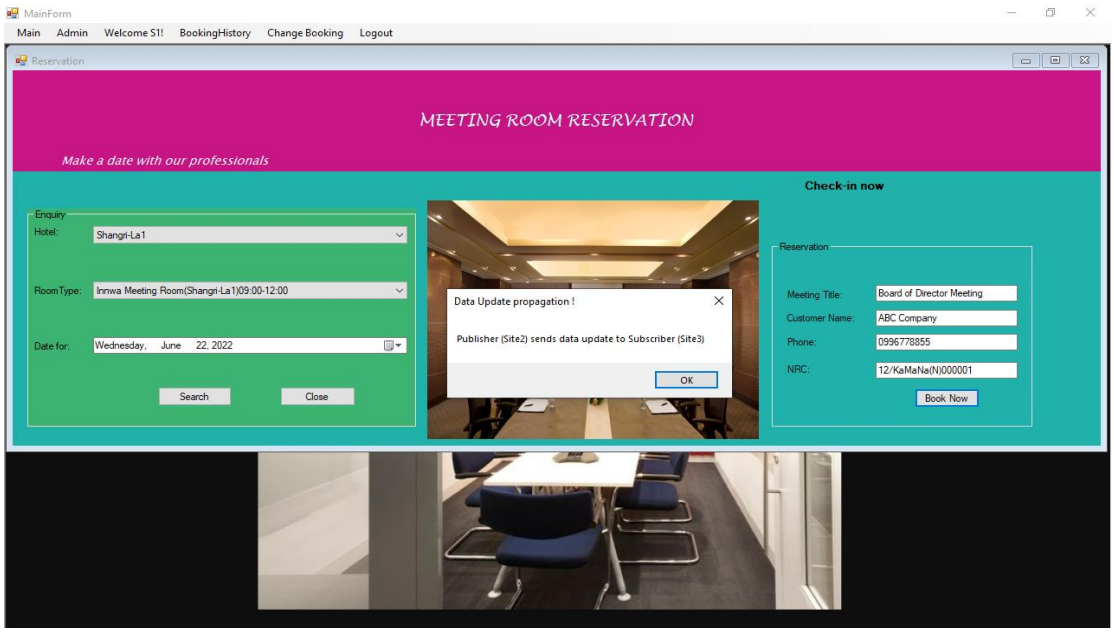

Figure 4.8 Data Update Propagation by Site 2 to Site 3

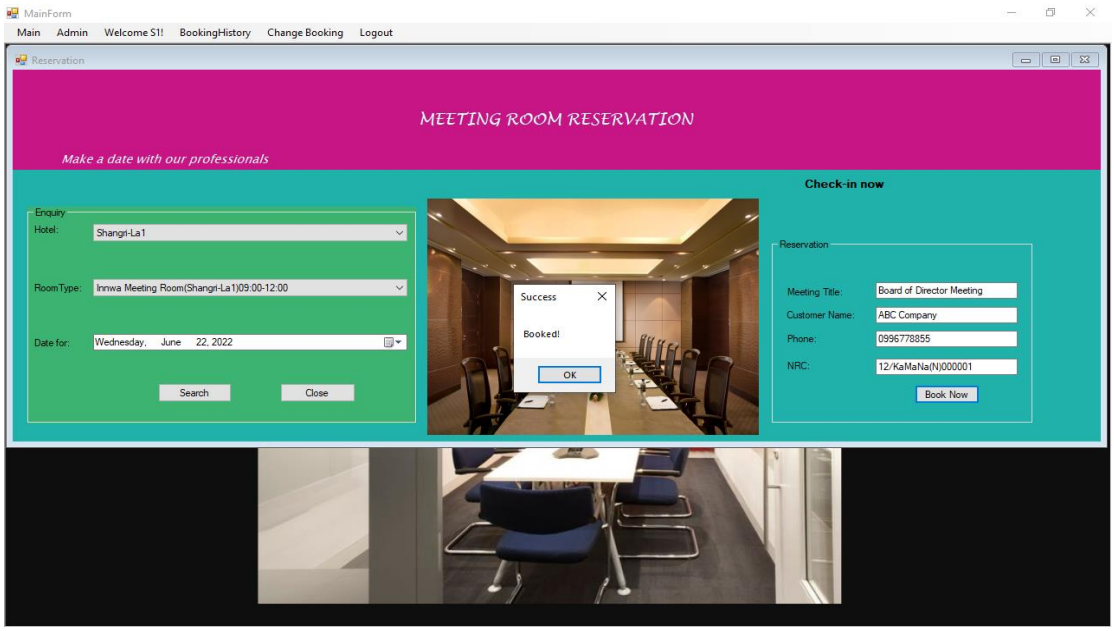

Figure 4.9 Notification for Booking Success

When a Meeting Organizer wants to book an event, the process makes requests to all other processes and waits for their replies. After getting all reply, the process decreases the number of room vacancies by 1. Then its multicasts the updated information to all other processes. System calls the event service to notify () operation, which is then responsible for routing the event through the event broker network to the interested users. Step-by-step operational instructions on how to reserve and manage meeting room requests are published and maintained separately by the collection of independent components located on different machines that share messages with each other. Meeting Organizers checked the room's status prior to scheduling the room to avoid double-booking. All meetings are subject to relocation due to importance of the meeting, location, and time of scheduled meeting. The system will make all reasonable attempts to provide as much advanced notice as possible to the Meeting Organizer and relocate the meeting to a comparable room when possible. In Figure 4.7 and 4.8, the data propagation approach is described, and in Figure 4.9, this system will be displayed booking success information.

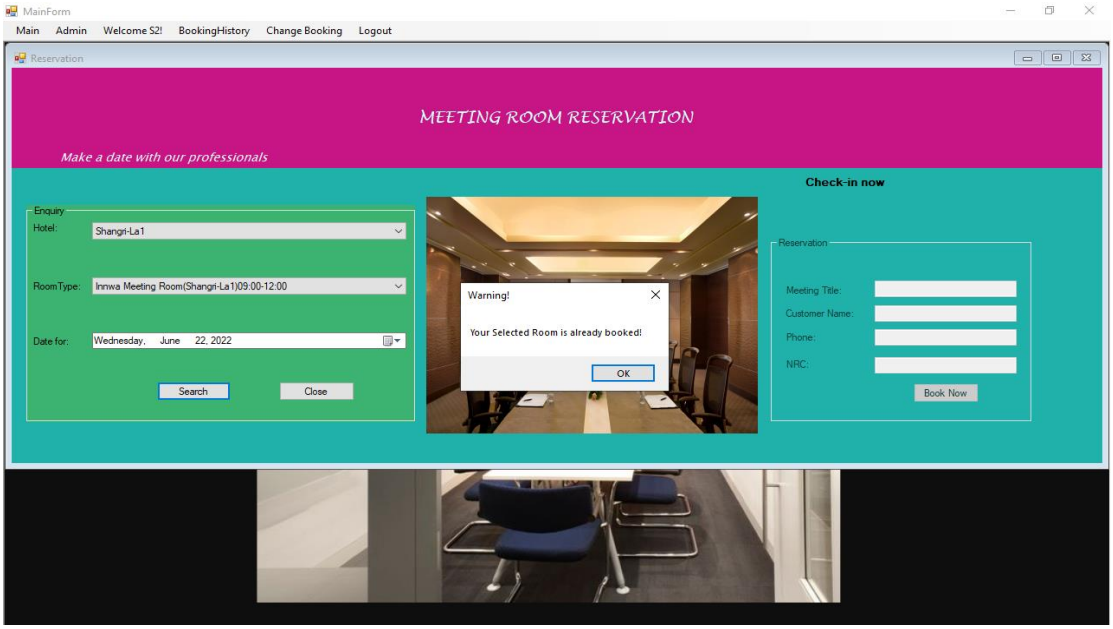

Figure 4.10 Restriction to avoid double-booking

After the booking success, the system will notify that the room is no longer available if someone else booked it, the warning message as shown in Figure 4.10.

4.8 Change Booking Process

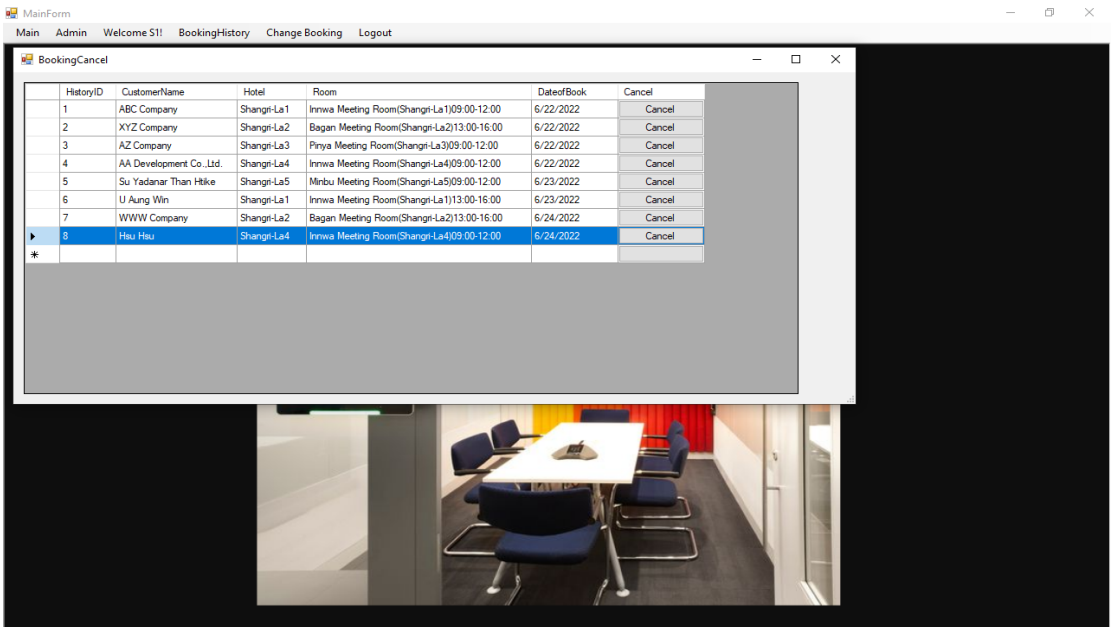

Figure 4.11 Booking Cancellation

If the customer wants to reschedule the meeting for various reasons, Meeting Organizer is required to cancel the originally booked meeting before the new meeting can be booked. This system will be showed Booking History for cancellation as shown in Figure 4.11.

4.9 Concurrency Controlling

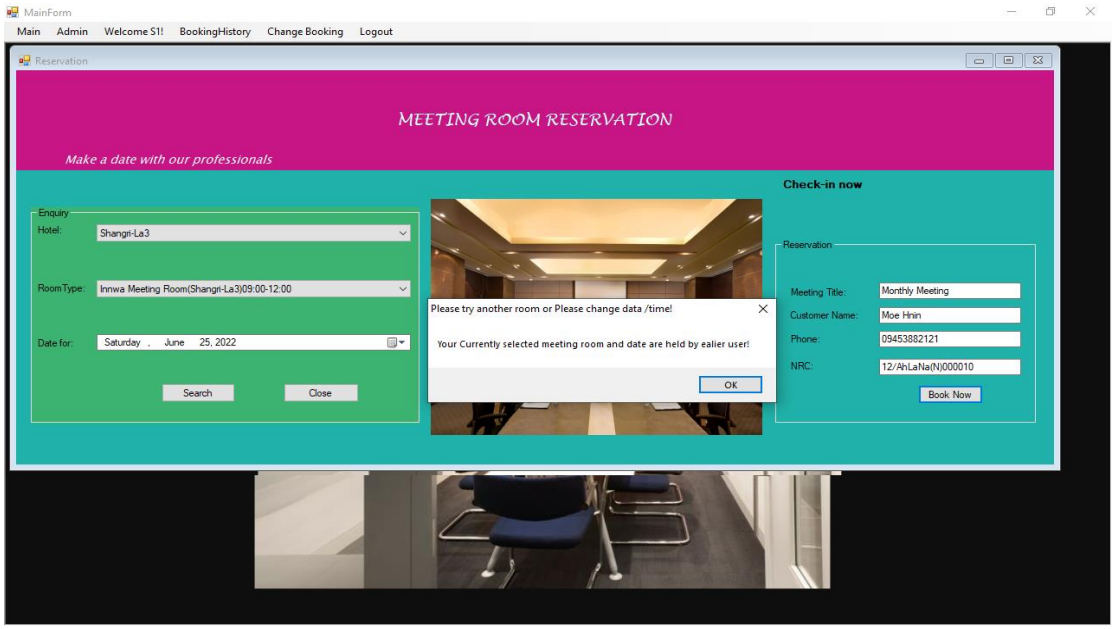

Figure 4.12 Notification for Concurrent Accessing on same event

This system controls not only multi-level user access but also simultaneous access to the same attribute. As shown in Figure 4.12, if the same event is requested concurrently more than one user, the system will reject it by showing a notification that the late user is no longer available. After committing to early user transactions, data updates are sent to the relevant user for the meeting room information updated by the event broker.

CHAPTER 5

CONCLUSION

This system offers a way to enhance the publish-subscribe model by connecting message brokers with peer-to-peer overlay networks. Publish-subscribe provides an effective model of communication by loose decoupling the link between subscribers and publishers. The overlay culture of connecting with brokers is an important part of this system. The meeting room booking management system provides all clients' accessibility, accuracy and continuity to the records by controlling the scribe to event-based routing.

5.1 Advantages of the System

i. This system allows Meeting Organizer to effortlessly manage meetings across all rooms among hotel outlets, without interruption.
ii. The present detection feature makes sure that the meeting room booking systems status updates to "booked" across all the counters.
iii. The meeting rooms occupancy detection solutions provide accurate, processing streaming data in real time on how and when different rooms are being used.
iv. The program can be configured, monitored and displayed to manage meeting rooms.
v. The concurrent problem found in Space decoupling and Time decoupling can be avoided in the proposed system.

5.2 Limitations of the System

The proposed system has some limitations. The first limitation of the system is that the system is based on peer-to-peer distributed network. Once the quantity of devices connecting the network increases, there will be a performance declined since each computer is being accessed by other users. As a result, P2P networks do not work well with growing networks. The second limitation is that P2P networks do not have centralized data that makes it difficult to back up data. Therefore, needs to make a separate backup of the data stored on each computer. The third limitation is that

maximum hotel set size is only 5. The fourth limitation is that the proposed system will not be accepted by online mobile and web.

5.3 Further Extensions

In this thesis, the proposed system for the future work could be considered the supply of more middleware services like composite event detection, persistent events, access control, transactions and support for mobile event customers who are partially connected with larger data sets and a higher workload to obtain better performance and scalability statistics. Additionally, it’s necessary to analyze ways of dynamically changing the overlay network configuration in response to the distribution of subscriptions, advertisements and events.

AUTHOR'S PUBLICATION

[1] Su Yadanar Than Htike, San Thida, "Meeting Room Management System on Peer-to-Peer overlay network by scribe to Event-Based routing", Parallel and Soft Computing Journal (PSC), UCSY, Yangon, Myanmar, Feb 2020.

REFERENCES

[1] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. “Design and evaluation of a wide-area event notification service, ACM Transactions on Computer Systems”, 19(3):332–383, 2001.

[2] G. Cugola, E. D. Nitto, and A. Fuggetta, “The jedi event-based infrastructure and its application to the development of the opss wfms”. IEEE Trans. Softw. Eng., 27(9):827–850, 2001.

[3] G. Mühl. “Large-Scale Content-Based Publish/Subscribe Systems”, PhD thesis, 2003.

[4] G. Perng, C. Wang, and M. Reiter, “Providing content-based services in a peer-to-peer environment”. In DEBS ’04: “Proceedings of the 2nd international workshop on Distributed event-based systems”, pages 74–79. ACM Press, 2004.

[5] L. F. Cabrera, M. B. Jones, and M. Theimer. Herald, “Achieving a global event notification service”, pages 87–94, 2001.

[6] M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron, “SCRIBE: A large-scale and decentralized application-level multicast infrastructure”, IEEE Journal on Selected Areas in Communications (JSAC), 20(8):100–110, 2002.

[7] M. Castro, P. Druschel, Y. Hu, and A. Rowstron, “Topologyaware routing in structured peer-to-peer overlay networks”, 2002.

[8] M. Castro. “An evaluation of scalable application-level multicast built using peer-to-peer overlay networks”, 2003.

[9] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, “The many faces of publish/subscribe”, ACM Comput. Surv., 35(2):114–131, 2003.

[10] Su Su Aung, “Event Notifications among Distributed Objects in Dealing Room System”, M.C.Sc. 2012, University of Computer Studies, Yangon.

[11] Thida Kyaw, “Monitoring the Number of Vacancies in a Car Parking by Using Distributed Mutual Exclusion”, M.C.Sc. 2011, University of Computer Studies, Yangon.