AAU

October 2008

Mobile Agents for Multi-Level Transaction of Distributed Multimedia Database

A Thesis submitted to the School of Graduate Studies of Addis Ababa University, in partial fulfillment of the requirements for the Degree of Master of Science in Computer Engineering.

By: Betiglu Mengistu Nigatu

School of Graduate Studies

Addis Ababa University

Mobile Agents for Multi-Level Transaction of Distributed Multimedia Database

A Thesis submitted to the School of Graduate Studies of Addis Ababa University, in partial fulfillment of the requirements for the Degree of Master of Science in Computer Engineering.

By: Betiglu Mengistu Nigatu

Advisor: Prof. Sayed Nouh

Addis Ababa

October 2008

School of Graduate Studies

Addis Ababa University

Mobile Agents for Multi-Level Transaction of Distributed Multimedia Database

By: Betiglu Mengistu Nigatu

Approval by Board of Examiners

Dr. Mengesha Mamo	
Chairman, Department of Electrical and Computer Engineering	Signature
Prof. Sayed Nouh	
Advisor	Signature
Dr. Mulugeta Libse	
External Examiner	Signature
Dr. Kuhmuda Raymond	
Internal Examiner	Signature

Acknowledgment

I am so grateful for my advisor Prof. Sayed Nouh for enlightening me to this wonderful Thesis work. From the day of inception to the last day your enthusiastic advise were great asset.

I am indebted to all the staff members of the Department for their material and moral support throughout the time of this research work. All of your comments and advises were valuable and greatly appreciated. Dr. Kuhmuda takes the lion share in this regard as your feedbacks on every seminar were indispensable. Kebre and Sintish, I can never be grateful enough for all the push and care, without you I would have been a dropout sometime back.

Numerous friends and colleagues are on the line to be duly acknowledged; though I cannot mention all, few in alphabetical order are: Abraham, Azeb, Bante, Biniam, Dereje, Fitsum, Hale, Kiros, Lebsework, Mesfin and Mubarek. All deserve my gratitude in every aspect of their effort and support. Those encouraging smiles and share of ideas were driving me to success.

Finally, but not least my family have always been my aspirations to keep me going with their endless love and care.

"I owe everyone one of you in my circle for your indulgent and encouraging support. One way or the other I am indebted to all of you."

Betiglu

Abstract

In today's information transfer, multimedia is playing a major role and it has vast and rigorous issues that can be studied. Its nature is widely varied and the objects are characterized in various ways requiring a fully developed multimedia database management system to support the current diverse applications. Most of the operations in multimedia application are data intensive; hence it is advantageous to make use of distributed applications in this regard. Also having a distributed multimedia database is indispensable for the motto "***Multimedia Anywhere, Anytime***". A multimedia database management system needs to address complex design issues such as: composition and decomposition of multimedia objects; operations of multimedia objects with media synchronization; security; consistency, atomicity, and concurrent access in a distributed environment; and long transactions and multi-level/nested transactions. To accomplish this level of interaction, an efficient information, communication, and cooperation environment is crucial.

In this thesis work, we present mobile agent-based architecture for managing distributed multimedia databases' multi-level transactions. The case study considers decomposition and composition transactions in media objects uploading and downloading request of a multimedia database. The transactions are studied in three layers: index, data and presentation with the use of concurrent executions of agents in an agent platform middleware. The middleware is designed to run on every node of the distributed system outfitted with standard relational database management system for the binary large objects storage of the multimedia.

The performance of the system/architecture is then tested and analyzed for a prototype over distributed computers in the Intranet of the faculty for efficient information distribution and retrieval. Its performance is then compared with a sample legacy application build from the ordinary remote procedure call execution of the multi-level transactions to meet the temporal quality of service requirement of media objects uploading and downloading.

KEYWORDS

Composition, Database, Decomposition, Distributed, Download, Middleware, Mobile Agent, Multimedia, Platform, Transaction, Upload

Table of Contents

List of Figures

Acronyms

ACID – Atomicity, Consistency, Isolation and Durability

AP – Agent Platform

APA – Arrival Portal Agent

API – Application Program Interface

APSL – Agent Programming System and Language

ASA – Agents' System Architecture

AVI – Audio Video Interleaved

BLOB – Binary Large OBject

bps – Bits per second

Bps – Bytes per second

CORBA – Common Object Request Broker Architecture

DB – Database

DBA – Database Administrator

DBMS – Database Management System

DCOM – Distributed COM

DDB – Distributed Database

DDBMS – Distributed Database Management System

DDBS – Distributed Database System

DES – Data Encryption Standard

DIM – Distributed Information Management

DMMDB – Distributed Multimedia Database

DMMDBS – Distributed Multimedia Database System

DPA – Departure Portal Agent

FTP – File Transfer Protocol

HDD – Hard Disk Drive

JMF – Java Media Framework

MA – Mobile Agent

MAP – Mobile Agent Platform

MLT – Multi-Level Transaction

MMDB – Multimedia Database

MMDBMS – Multimedia Database Management System

MPEG – Moving Pictures Expert Group

OO – Object-Oriented

P2P – Peer-to-Peer

PA – Portal Agent

QoS – Quality of Service

RA – Resource Agent

RAM – Random Access Memory

RDBMS – Relational Database Management System

RMI – Remote Method Invocation

RPC – Remote Procedure Call

TA – Transactional Agent

WWW – World Wide Web

1 Introduction

1.1 Background

In its vital usage of today's information era, multimedia has wide variety of nature and application. The nature of the multimedia can also be characterized in different ways as its huge volume and resource demanding applications.

1.1.1 Nature of Multimedia

The nature of multimedia data and information management systems require them to manage data that are manifested in diverse forms such as text, images, voice, graphics, and video (Adjeroh and Nwosu July-September 1997).

Most of these multimedia data type's nature demands temporal requirements, which have implications on their storage, manipulation, and presentation. Such multimedia objects as video, audio and animation sequences are incorporated with time based responses that need to be addressed in the design of the middleware for persistent data presentation. The problems become more acute when various data types from possibly distributed sources must be presented within or at a given time (Adjeroh and Nwosu July-September 1997).

Besides, their huge volumes of data and temporal requirements that characterize multimedia information, the most important characteristics of multimedia objects can be summarized as follows:

Multimedia objects are complex: due to their natural requirement for large capacity and temporal demand, multimedia objects are complex for storage as well as presentation, therefore less than completely captured in a database management system.

Multimedia objects are audiovisual in nature: different applications that utilize multimedia data often are required to manage audiovisual objects that are amenable to multiple interpretations.

Multimedia objects are context sensitive: integration of multimedia objects into heterogeneous systems demands for accessing the resource on their context. Hence, queries looking for multimedia objects need for context-

sensitive multimedia objects that are likely to use subjective descriptors that are at best fuzzy in their interpretation.

Multimedia objects may be included in fuzzy classes: the nature of multimedia is also characterized on their uncertainty which the fuzzy object oriented data (FOOD) model may be used, for storing and manipulating the objects. This mapping is done in a way that it preserves most of the information presented at the conceptual level (Aygün, Yazici and Arica 1998).

At the very least, the characteristics of multimedia objects require them to be adaptable in terms of resource usage, such as the use of screen space, network bandwidth, and computational resources. Multimedia objects should also be able to alter their modes of representation in response to the demand of the users' applications.

1.1.2 Multimedia Applications

Multimedia data are used in different applications that can be broadly categorized based on their characteristics as:

Repository Application: the main use is for storing large amount of multimedia data and metadata for retrieval purpose. Central repository database system with hierarchy of storage may maintain such applications. Examples in here include: satellite images, engineering drawings, space photos, and radiology scanned pictures.

Presentation Application: large amount of applications involve delivery of multimedia data subjected to temporal constraint. The main issue is delivery of multimedia data with the concern of data rate quality of service (QoS). Data is consumed as it is delivered, unlike in repository applications where it may be processed later (e.g. multimedia electronic email, video/audio conferencing).

Collaborative Work using Multimedia Information: complex engineering design tasks that may require merging drawings, fitting subjects to design constraints and generating new documentation, change notifications and so forth. Examples in here include: Telemedicine – doctors collaborating among themselves; Engineers sharing design templates and drawings.

In general, all the applications require a data storage and management system for backing up their services. In cases, where the applications are to be used by multiple users distributed across a network, the management system may as well be distributed. Thus, the database managing the multimedia data may be required to be distributed with a probability of heterogeneous systems.

In regard to the nature and characteristics of multimedia objects number of research areas can be pointed out. One typical area is extending the current mobile agent-based systems to support the middleware requirements of multimedia database system. Such a middleware addresses related issues in the multimedia domain (such as bandwidth allocation, reducing network traffic, and initializing and gathering distributed resources in multimedia communication.)

Some of the other issues that can also be listed includes: information searching and filtering in distributed database systems (DDBS), consistency and integrity of data in DDBS, security of distributed database management system (DDBMS), composition and decomposition of multimedia data, persistency of multimedia data, multi-level transaction (MLT) managements in DDBMS, and content-based retrieval of multimedia information.

Information searching and filtering in DDBS may be driven by proliferation of data and heterogeneous environments, that brings challenge to protect data from flooding the network client DBMS. It is required to introduce smart, lightweight, flexible independent and portable client programs. This leads into research areas for the development of services and utilities for wide variety of distributed information search and retrieval with mobile agents.

1.1.3 Mobile Agents

Mobile agent (MA) is a software abstraction (program) that exhibits features of autonomy, social ability, learning, and most importantly, mobility. MAs traverse over a network, often on behalf of their users to access information in dynamic heterogeneous environments and return back to the user with their result. The agents are managed and monitored by a platform known as mobile agent platform (MAP). The platform is responsible for monitoring the agents' behavior, creation and disposal of agents, and agents transfer over a network. It facilitates an environment for executing, migrating, suspending, resuming and communicating agents.

Their mobility and lightweight network resource utilization make MA technology a good programming paradigm for building agile distributed systems. Hence, MAs can be helpful in various applications related to distributed multimedia system. In multimedia retrieval scenario, they are used to access information across the Intranet/Internet: information stored in a remote multimedia database (de Groot, et al. 2005).

1.2 Statement of the Problem

In the age of proliferating information and communication systems, information is becoming a vital production and service provision factor. More and more multimedia data is now being stored on the web and effective management of this data is becoming a critical need. The interaction between the different categories of users is getting to be supported thoroughly by multimedia, and without intensive communication and cooperation, this interaction is hardly manageable.

As mentioned in section 1.1.1; the very nature of multimedia data requires it to be large in size while demanding real time processing for presentation to the user. Though, the technological advancement in the storage capacity of memory devices such as RAM and HDD is rapid, multimedia applications are growing at the same pace in their resource utilization. For the storage and presentation reasons, multimedia applications are distributed over multiple sites. Thus, requiring a DDBMS that manages the transactions of the system from storage to presentation. Such systems need to preserve and present the multimedia data efficiently within the available resource. The transactions in multimedia supporting management systems are complex and multi-level in nature. We also need to ensure that the data is protected from unauthorized access and malicious corruption during storage as well as communication.

To accomplish the required level of interaction and maintain data consistency and security, an efficient information, communication, and cooperation environment is crucial. The realization of such an environment requires a middleware that provides the necessary multimedia services for the cooperative applications and that also abstracts network and delivery details. The middleware also needs to manage the complex and MLTs of the DDBMS.

1.3 Rationale

From the use of multimedia in various applications and the fact that pictures are worth of thousands of words multimedia objects are becoming necessities in day to day activities. For this and more the Moto behind this thesis work is the currently widespread phrase "***Multimedia Anywhere Anytime***".

In information dissemination, multimedia data are playing major roles and they have vast and rigorous issues that can be studied. The acquisition, generation, storage and processing of multimedia data in computers and transmission over networks have grown tremendously in the recent past. This astonishing growth is made possible by three factors.

a. Personal computers usage becomes widespread and their computational power gets increased. Also technological advancements resulted in high-resolution devices, which can capture and display multimedia data (digital cameras, scanners, monitors, and printers). Also there came high-density storage devices.

b. High-speed data communication networks are available nowadays. The Web has wildly proliferated and software for manipulating multimedia data is now available.

c. Some specific existing applications and future applications need to live with multimedia data. This trend is expected to go up in the days to come.

These advents of multimedia computing and information processing have necessitated the need for multimedia database management systems (MMDBMSs). An MMDBMS is then required to provide efficient storage and manipulation of the multimedia data.

To provide the various functionalities of multimedia applications the database management system performs MLTs that may be done over a distributed system. An important factor for the design of a distributed MMDBMS is today's mobile society, where access to multimedia services "***anytime, anywhere***" is becoming increasingly important. A simple distributed MMDBMS application requires a lot of metadata to be searched and exchanged transactions. The multimedia database (MMDB) must contain semantic information on the movies, such as the name of

the actors playing in the movie, the movie genre, and so on. It must provide a video summarization that matches the user preferences in terms of content as well as delivery format.

If a middleware is instead used to handle the necessary transaction of the DBMS it would provide scalability and platform independency. However, the middleware is required to be less resource demanding architecture to fulfill temporal requirements of multimedia objects. Thus, MA architectures are worth to be considered to provide the lightweight process in network bandwidth utilization of multimedia applications.

1.4 Objective of the Thesis

In regards to the need of a lightweight middleware to support the requirements of DMMDB transactions, this Thesis work is presented to address the following general and specific objectives.

General Objective

In general, the Thesis addresses the issue of MA based architecture for MLT management in DMMDB. Hence, the overall purpose of this research work is to design and study the MA based architecture to include the features required by DMMDB transactions.

Specific Objective

As an immediate objective it focuses on the MLT transactions that are encountered in media object uploading and downloading. The MA architecture in DMMDB will be studied and designed to allow distributed access, and concurrent querying over multiple and heterogeneous multimedia systems in a modular and scalable fashion. The MLTs considered are the ones that are required in media object decomposition and composition during uploading and downloading of the media file.

The MA based middleware is studied to handle the following technical issues in the MLT of a DMMDB:

Retrieval and presentation: multimedia objects are required to be located and presented to the user from the storage management. The process of retrieval

and presentation then requires transactions at different level of the system such as data level, index level and presentation level.

Identity management: identifying an agent in such a system is crucial as it would be valuable for the purpose of administrating which entities (users, agents, services, and locations) are in the system; and also for the application of access policies to determine who is allowed to do what and with which credentials.

Integrity and consistency: to ensure correct functioning of an MMDB, the integrity of both the agent platform and visiting agents is essential. In other words, it is crucial that the data and transmissions are not unduly altered, erased or supplemented and that the physical objects involved are not damaged or destroyed. As well the data distributed need to be maintained without disturbing the consistency of the system.

Logging and tracing: logging communication and actions of entities in the system to reconstruct an agent's interactions is particularly useful if damage is incurred. It is needed to trace errors in the system and determine liability.

1.5 Thesis Organization

The remainder of this Thesis's report for the research work is organized into the following six chapters:

- **Chapter 2** of the report presents a brief overview of distributed database and MMDB technologies and applications. It studies the background of the problem to address the points that need to be covered in the design of the proposed architecture.

- **Chapter 3** then discusses the use of MAs in distributed applications. It points out the advantages and disadvantages of an MA-based system in a distributed environment. The section also reviews the use of MAs for multimedia applications and shortly summarizes related works in this area.

- Under **chapter 4**, we will discuss the proposed architecture of the agent-based system for the MLT of a distributed multimedia database (DMMDB). The section explains the major functionalities supported in the

architecture for the two main operations of multimedia application data saving and retrieval.

- The materials and methods used in the development of the prototype for the designed architecture are explained and summarized under **chapter 5** of this report.

- **Chapter 6** analyzes and discusses the results of the tests performed on the proposed architecture prototype and its comparison with a legacy remote procedure call (RPC) application.

- Finally, conclusions and recommendations for future works are presented in **chapter 7**.

1.6 Limitations

As mentioned in the background of this chapter, there are a number of areas that can be considered for research in the area of distributed multimedia data. However, the reason behind this thesis work is the high resource demand of multimedia presentation. Hence, only limited issues of QoS are studied with the design of MA-based architecture for the MLT of DMMDB.

The scope of this work is limited in studying the proposed architecture for multimedia objects composition and decomposition that may be required during object uploading and downloading (retrieval) over distributed sites. The architecture is tested only for a video file that can also be generalized for any media object with the decomposition and composition modules customized for the media. The composition and decompositions considered in the study are restricted to binary level operations based on the size of a media object. However, further logical compositions and decompositions can easily be incorporated to the system.

2 Distributed Multimedia Database

In this chapter, we will survey literatures in DDBSs and multimedia systems that are helpful in the design of the proposed architecture for managing transactions of DMMDB. Designing and architectural issues are broadly the focus of the chapter.

2.1 Distributed Database

Literatures have given various definitions of distributed systems, but in common understanding a distributed system is regarded as collection of independent computers that appears to its users as a single coherent system. A distributed database (DDB) is then a collection of multiple, logically interrelated databases (DBs) distributed over a computer network for repository of related data as shown in Figure 2-1.

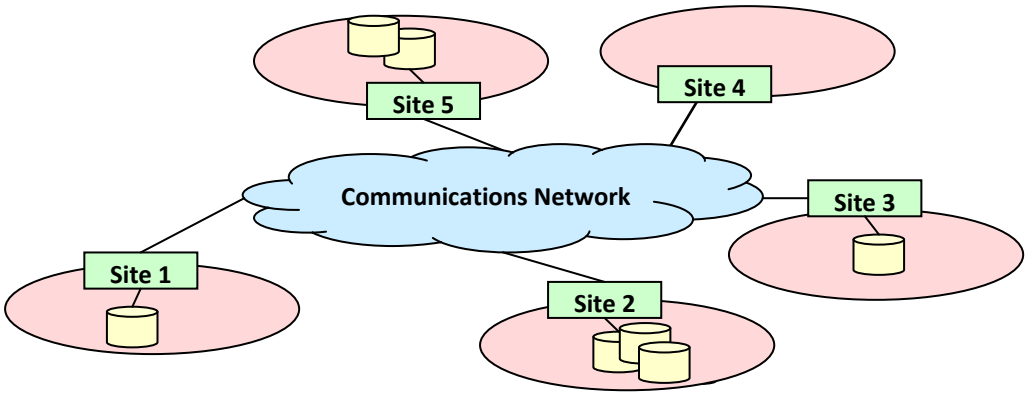


Figure 2-1: Distributed database system (DDBS)

The ***sites*** (also interchangeably referred to as ***nodes***) in the DDBS may be equipped with database management system (DBMS) for manipulating and managing the data on their location. These DBMSs are distributed over the sites but need to be presented to the user as a single integrated unit to provide the distributed abstraction.

DDBMS is the software system that permits the management of the distributed database and makes the distribution transparent to the users. The term distributed database system is typically used to refer to the combination of DDB and the DDBMS.

2.1.1 Architecture of Distributed Database System

DDBS consists of several DBs distributed over several sites. There are a number of alternative architectures for DDB systems. Figure 2-2 depicts the various architectures and their hierarchy (Piattini and Díaz 2000).

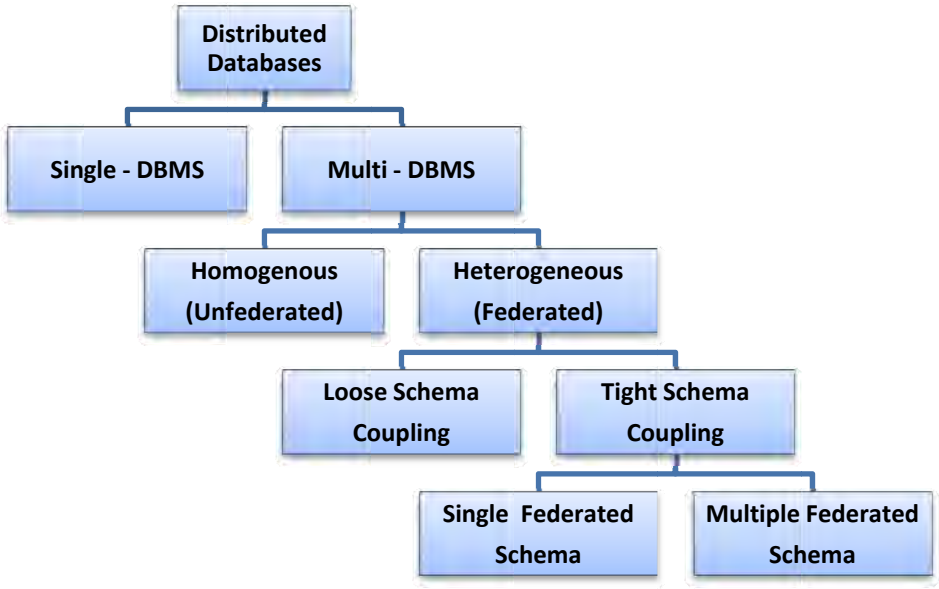


Figure 2-2: Distributed database system (DDBS) architecture

Single – DBMS architecture: is an architecture where access to the DBs is controlled by a single DBMS process.

Multi – DBMS architecture: unlike the single – DBMS architecture the multi – DBMS architecture has several independent DBMS processes, each controlling access to its own local DB. The architecture is of two types as:

- ***Homogenous (unfederated) architecture***: where there is a single DB administration authority that decides what information is stored in each DB, how the information is stored and accessed, and who is able to access.

- ***Heterogeneous (federated) architecture***: in this architecture the DB administration authority is divided between the databases administrators (DBAs) of each local DB (the local DBAs) and the DBAs for the overall federation (the global DBAs). The local DBAs have complete authority over the information in their DBs, and what part of that information is made available to the federation, that is, to global queries and transactions. This information is represented in the form of

one or more export schemas for each local DB. The global DBAs control global access to the system, but must accept the access restrictions imposed by the local DBAs. The heterogeneous architecture has two types of schema: *loosely coupled* and *tightly coupled* schemas.

- ***Loosely coupled schema***: A federated DDB is loosely coupled if there is no global schema provided by the global database administrator (DBA), and it is the users' responsibility to define the global schemas required to support their applications.

- ***Tightly coupled schema***: A federated DDB is said to be tightly coupled if the global DBAs maintain one or more global schemas that provide an integrated view through which global queries and transactions can access the information stored in the local DBs. These present extra difficulty to provide an integrated view of the information stored in the local DBs. The two schemes in this category are: ***single federated*** that refers for the single schema and ***multiple federated*** referring to multiple schemes.

2.1.2 Pros and Cons of Distributed Database System

The various architectures of a distributed system are having their special features that make them advantageous of one another. But generally speaking, some of the advantages of DDBS can be summarized as follows.

i. **Management of distributed data with different level of transparency**: that refers to the sense of hiding the details of where each file (object, table) is physically stored within the system. Transparency in a distributed data is implemented in terms of:

   - ***Distribution or network transparency***: a freedom of the user from the operation details of networks. Network transparency allows the user to name the objects in the system unambiguously of their location without any additional specification. This ability of naming the object unambiguously is known as ***Naming transparency.*** Where as to have commands for the operations irrespective of location is known as ***Location transparency***.

- ***Replication transparency***: allows the user to be unaware of the replicated instance of data if it exists.

- ***Fragmentation transparency***: whenever there is fragmentation in the system, fragmentation transparency keeps the user unaware of the existence for fragmentation.

ii. **Increased reliability and availability**: reliability asserts that the system runs with no down at any certain time and availability assert the existence of the system for all time.

iii. **Improved performance**: proximity of data to its points of use (***data localization***) allows for the local transactions accessing data closer. This requires some support for fragmentation and replication which in turn allows parallelism in execution at levels of ***interquery parallelism*** and ***intraquery parallelism***. In interquery parallelism queries or transactions are executed in parallel. On the other hand, Intraquery parallelism refers to executing a single query or transaction in parallel on multiple processors and disks. While the primary use of interquery parallelism is to *scale up* transaction processing intraquery parallelism speeds up running transaction.

iv. **Scalability**: easy of expansion in terms of adding more data, increasing database sizes, or adding more units is also one other advantage of DDBS.

The use of DDBS provides the advantages sited above with the cost of ***complexity***. The system adds complexity in handling fragmentation and replication, concurrent transactions, and so forth.

2.2 Distributed Database Management System

As it is seen in the previous section a distributed system in general provides various advantages that can also be employed in a database management system. Besides that, the current technology of inter-networked computer calls for distributed systems let alone the current need for distributed information dissemination.

2.2.1 Design Issues of Distributed Database Management Systems

A DDBMS is expected to provide all the functionalities of ordinary database management systems plus it needs to coordinate the various sites that are working in the system. Hence, in the design of DDBMS there are various issues to be considered some of them include:

- ***Keeping truck of data*** distribution, fragmentation, and replication.
- ***Distributed query processing*** ability to access remote sites and transmit queries among the various sites via the communication network.
- ***Distributed transaction management*** to handle queries and transactions that access data from more than one site and to synchronize the access to distributed data and maintain integrity of the overall database.
- ***Replicated data management*** to decide which copy of a replicated data item to access and maintain the consistency of copies.
- ***Distributed database recovery*** from a site crash or communication network failure.
- ***Distributed directory (catalog) management*** managing a directory that contains the metadata of the system globally for the entire distributed database or at every local site.
- ***Data security management*** for data authorization and/or access privileges of users.

2.2.2 Transaction in Distributed Database

The transaction management in DDBS deals with the issues of: Queries and Retrieval, Concurrency, Security and Performance.

Queries and Retrieval: refer to processing of global queries in the DDB. The quires are optimized generating a set of alternative query plans, estimating the cost of each plan, and selecting the cheapest plan for execution. Selected query is processed based on local information about access paths and DB statistics returning the result set (if any) of the query.

Concurrency: the DDBS transaction manager coordinates concurrently executing transactions so as to guarantee the so-called ACID (Atomicity, Consistency, Isolation and Durability) properties:

- ***Atomicity***: Either all or none of a transaction is executed,
- ***Consistency***: Transactions must leave the data in a consistent state that is satisfying all the stated integrity constraints,
- ***Isolation***: It must appear to users that transactions are being executed one after the other, even though they may be interleaved, and
- ***Durability***: If a transaction has committed, its effects must not be lost.

Then the purpose of the distributed transaction management is to maintain the ACID properties of global transactions, that is, transactions expressed with respect to the global or external schemas of the DDB system. The two mechanisms by which a transaction manager guarantees the ACID properties are ***Concurrency control*** and ***Recovery***. The concurrency control uses various locking strategies to ensure consistency and isolation; whereas, recovery algorithms are used to ensure atomicity and durability.

Security: identity and permission levels have to be monitored for transactions requesting access and/or modification of shared resource from a DB in a distributed environment. This requirement calls for secured transaction management systems for the database.

Performance: one way for measuring the performance of transaction in a distributed database could be the response time and resource utilization of the transactions within their environments. The transaction management system needs to deliver the required task satisfying the performance requirement of the system.

2.3 Multimedia Database

The huge amount of data in different multimedia-related applications warranted to have databases as they provide consistency, concurrency, integrity, security and availability of data. From a user perspective, databases provide functionalities for easy manipulation, query and retrieval of highly relevant information from huge collections of stored data.

2.3.1 Multimedia Database Application

Multimedia data are blessed with a number of exciting features. Recently, almost in every day to day activity we are encountering the use of interlinked computers that provides us with the information we are looking for in more attractive and intuitive way. They can provide more effective dissemination of information in science, engineering, medicine, modern biology, social sciences, and in many more. They also facilitate the development of new paradigms in distance learning and interactive personal and group entertainment. As the nature of multimedia is wide and versatile it is being used in many more applications. Some of the applications can be summarized as:

- **MMDB for documents and record managements**: from the advent of the first digital system, the computer technology has shown an inspiring development and graphics and multimedia objects are becoming efficient way of documentation and record management. Multimedia database systems then can be applicable to manage the multimedia documentation and management systems.

- **MMDB for education and training**: in the current education system it is very common to use various multimedia objects to disseminate knowledge. Mentionable examples in this area include multimedia supported text books and training manuals that are prepared for self teaching. Also, in the e-learning system multimedia applications enjoy great share.

- **MMDB for marketing**: the current marketing system over the web is the highly favorable area where multimedia objects are used to attract customers as well present full information on advertising items. The use of multimedia objects for advertising benefits both parties as customers can also easily search for their interest from graphics and multimedia objects than textual presentation. Such data warehouses are supported with MMDBs.

- **MMDB for real time control and monitoring**: for process monitoring, security monitoring or even child care real time and nanny cameras are getting wide application. This application requires real time control and monitoring of multimedia data that may be backed up with an MMDB for storage.

From the summarized list and previous discussion it is clearly seen that the recent growth in using multimedia data has been phenomenal. Multimedia databases are essential for efficient management and effective use of huge amounts of multimedia data. However, the diversity of applications using multimedia data, the rapidly changing technology, and the inherent complexities in the semantic representation, interpretation and comparison for similarity pose many challenges in the development of MMDBMSs.

2.3.2 Distributed Multimedia Database Management System

The various application areas of MMDBMS require it to provide more facilities in addition to the traditional DBMS requirements. The major functionalities required by MMDBMS presented in (Shih 2003) include:

- Handling of various multimedia data types, and metadata,
- Handling of large amount of multimedia objects which are binary large objects (BLOBs) in nature,
- Providing high performance and cost effective storage management,
- Supporting database functions such as insert, delete, retrieve and update.

In order to support these functionalities various issues are considered in designing and construction of MMDBMS. Among the well known designing issues few as discussed in (Shih 2003) are:

Long transactions and multi-level/nested transactions: since multimedia objects often contain large amount of data, the transactions are time consuming and long in nature. And since multimedia objects usually exist in compound it is common that they form multi-level/nested instructions.

Composition and decomposition of multimedia objects: a multimedia object usually doesn't exist by itself instead it is accompanied with number of objects such as picture with a text and music. Composition and decomposition can be achieved with the use of an object-oriented (OO) approach with different types of links of various semantics.

Operations of multimedia objects with media synchronization: multimedia operations in composed objects require the synchronization of the objects such as: display pictures and text as well pictures and audio synchronously.

Persistence object: multimedia objects are often embedded with time constraining operations (such as video streaming), these multimedia operations with time constraint then should not affect the persistency of the multimedia objects.

Indexing and clustering: for fast accessing or locating of the multimedia objects indexing and clustering mechanisms are required.

Concurrent access and locking mechanisms for distributed computing: on distributed environment, one needs to consider concurrent access with a reasonable performance and locking mechanisms if the MMDBMS is to support computer supported cooperative work.

Security: if the multimedia is also to be available on a distributed environment over a network, security is an essential matter to consider.

Consistency, referential integrity, and error recovery: during data insertion, update and deletion, one needs to maintain the consistency of the objects' semantics in the MMDB. In case of exception due to both hardware and software failure, the MMDBMS should support recovery mechanism to its known consistent state.

Content-based multimedia information retrieval: unlike the traditional database searching schema text or numerical data comparison, MMDB requires complex searching and matching criteria which makes content-based information retrieval harder.

For an MMDB system to fully support the content-based multimedia system it needs to address the following points:

- ***Retrieval and Indexing***: semantically meaningful indexing methods must be developed and linked to low-level indexing and retrieval. However, it is hard to find any available content-based retrieval technique. Example: "find a person sitting and laughing in all the movies stored".

- ***Storage, Communication and Performance***: often multimedia data are of large amount hence may require compression techniques (adaptation of the digitally stored representation of the multimedia data) to satisfy the storage and bandwidth requirements of specific application. This leads into achieving the performance level satisfactory to the usage of the application.

- ***Streaming and Resource Scheduling***: continuous data presentation needs continuous data transportation and demands resource scheduling techniques assuming the QoS of the network and the servers are guaranteed.

- ***Presentation***: spatial and temporal constraints may be set on the presentation of data, hence leading to management of the presentation delivery in the multimedia servers.

Besides the aforementioned designing issues of MMDBMS, the distributed MMDBMS (DMMDBMS) needs to present the requirements of a DDBMSs discussed under section 2.2: *Keeping truck of data*, *Distributed query processing*, *Distributed transaction management*, *Replicated data management*, *Distributed database recovery*, *Distributed directory (catalog) management* and *Security* issues.

2.3.3 Architecture of Multimedia Database Management System

In multimedia delivering, streaming and receiving there have been variant types of architectures proposed. The two most common of the architectures used for the design of distributed multimedia database systems (DMMDBS) are:

1. **Client/Server Based Multimedia System**: Many distributed MMDBMSs (DMMDBMS), for pull applications rely on client/server architectures. In this scenario, client applications request multimedia data from the server, which are then processed locally. Obviously, this architecture minimizes the degree of parallel processing that can occur because of the synchronous nature of the request function. It is a common experience that pure client/server systems do not scale well with respect to the number of users, the heterogeneity of the terminals employed, and the size of the data requested. Figure 2-3 shows the architecture of a client/server system.

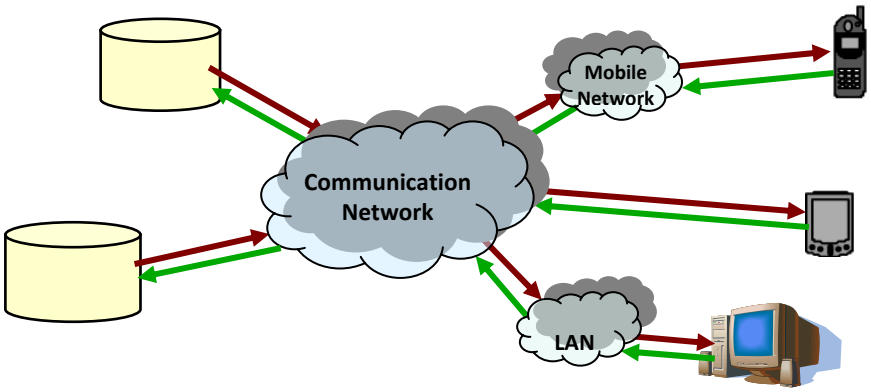


Figure 2-3: Client/server based multimedia system

2. **Peer-to-Peer System**: To allow greater flexibility, multimedia systems evolve from pure client/server system to peer-to-peer (P2P) systems with the notion of a "user", which is defined as an entity to consume and to produce digital multimedia items. P2P architecture is shown as in Figure 2-4, where portable devices with various capabilities send and transfer multimedia content from one device to another.

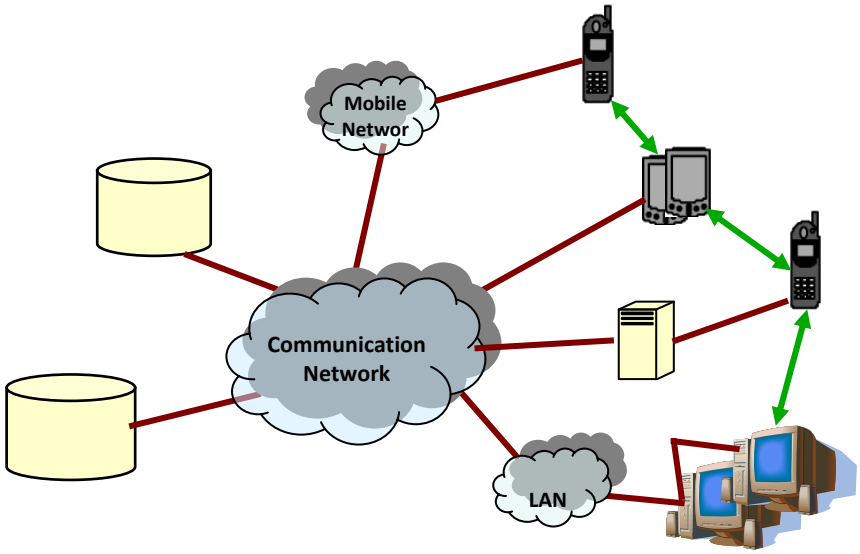


Figure 2-4: Peer-to-peer based multimedia system

The figure shows a PDA user, a mobile phone user, and PC users producing as well consuming content from the multimedia storage server but also communicating with each other, including making phone calls, sending e-mail, images, music, and videos to the other, although the terminal capability is quite different from PDAs to mobile phones or PCs.

Such a system must provide P2P services in some form; for instance, operators, which may have the following functions:

- To hold the content from the creation to the delivery,
- To check the capability of the other peer party and see whether it can process the multimedia data before delivering, and
- To create metadata to describe content received from the users.

Design of the management system for the MMDB can also be considered in various architectures, but the architecture usually contains three layers as an interface layer, object composition layer and storage layer.

i. **Interface layer:** a layer that deals with three tasks for interacting with the user:

   a) ***Object browsing***: this allows the user to find multimedia resource entities to be reused through queries.

   b) ***Query processing***: the queries, either text based or visualized, specify a number of conditions to the properties of resource and are processed to retrieve a list of candidate objects. Multimedia resources, unlike text or numerical information, cannot be effectively located using a text based query language. Even natural language presented in a text form is hard to precisely retrieve a picture or a video with certain content. ***Content-based information retrieval*** researches focus on the mechanism that allows the user to effectively find reusable multimedia objects, including pictures, sound, video, and other forms.

   c) ***Interaction of object composition/decomposition***: after the successful retrieval, the database interface should help the user to compose/decompose multimedia documents for presentation.

ii. ***Object composition layer***: the layer manages objects that require a number of links for composition as in object-oriented system (such as association links, similarity links, and inheritance links). These links are specified either via the database graphical user interface, or via a number of application program interface (API) functions.

iii. ***Storage layer***: handles storage of the multimedia data and the metadata. It deals with two performance related issues:

a) ***Clustering***: that organizes multimedia information physically on a hard disk for the system to be able to access the large binary data efficiently.

b) ***Indexing***: that is for fast locating of data essential to find the physical address of a multimedia object.

Usually, the performance of retrieval needs to guarantee some sort of QoS and to achieve multimedia synchronization.

2.4 Transactions in Multimedia Database

Unlike traditional database systems, which have text or numerical data, an MMDB or information system may contain different media objects. One of the inherent characteristics of multimedia data is the heavy time-dependence in that they are usually related by temporal relations which have to be maintained during their play out. Therefore, how to synchronize the various data types for sophisticated multimedia presentations is a major challenge. Transactions in DMMDB can thus have variant structures such as:

i. **Flat transaction**: Consisting of a sequence of primitive operations embraced between **begin** and **end** markers for the transaction.

ii. **Nested transaction**: The operations of a transaction may themselves be transactions. Creating nest of transactions that can be presented in hierarchy.

The nested transactions have the same properties as their parents hence may themselves have other nested transactions. Concurrency control and recovery concepts are also introduced within the transaction. Nested transaction can be of two types:

a) ***Closed nesting***: Sub-transactions begin after their parents and finish before them. Commitment of a sub-transaction is conditional upon the commitment of the parent (commitment through the root).

b) ***Open nesting***: Sub-transactions can execute and commit independently. Compensation may be necessary.

iii. **Multi-level transactions**: are variants of open nested transactions in which the sub-transactions correspond to operations at different levels of layered system architecture.

According to (Hasse and Weikum 1991) the point of MLTs is that the semantics of high-level operations can be exploited in order to increase concurrency. One typical example for the MLT of the MMDB is applied at two levels:

i. **Object Level**: semantic locks are dynamically acquired and held until end-of-transaction.

ii. **Page Level**: page locks are dynamically acquired during the execution of a sub-transaction and are released at the end-of sub-transaction.

An example used by (Hasse and Weikum 1991) for explaining the two level MLT is office document filing system where documents have a complex structure and can span many pages to be modified by the change() transaction. Figure 2-5 explains parallel execution of the two level transactions for two concurrent transactions T_1 and T_2 each having two sub-transactions T_{11} and T_{12}; and T_{21} and T_{22}, respectively.

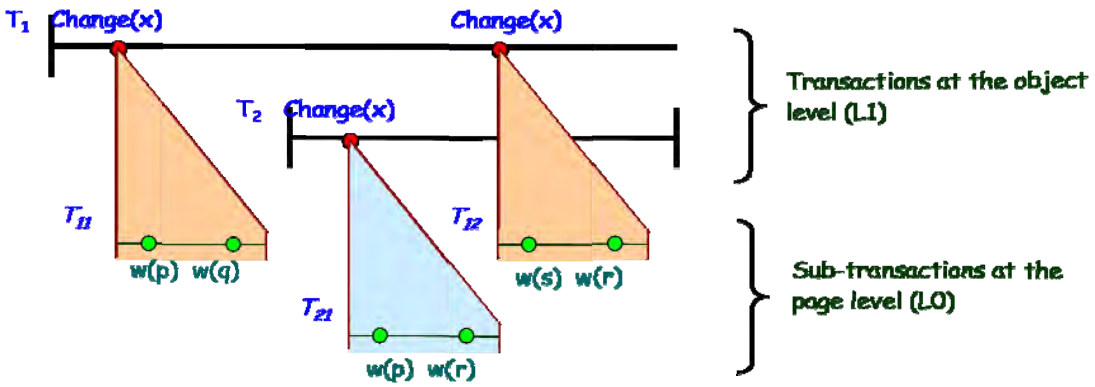


Figure 2-5: Parallel execution of Multi-level Transaction

The same MLT concept can be employed for the case of an MMDB where the levels can be considered to be data, index and presentation layers. Transactions in such layers may be executed concurrently as long as they do not affect the integrity and consistency of the system. The concurrency will be extended, if the database application is a distributed one.

3 Mobile Agents for Multimedia

From the previous chapter we have seen that MMDB is having tremendous application areas. It is also stated that it requires a middleware to support presentation and transaction management of the multimedia objects. In this chapter, we will study MA-based systems for distributed system in general and specifically to distributed multimedia applications.

3.1 State of the Art in Mobile Agents

Component-based software development technology is being used as a powerful approach for the development of distributed systems and multimedia systems. The technology can combine subcomponents into an application or a large-scale component. On the other hand, MAs are self-contained entities like software components and can travel from computer to computer under their own control. MA technology has received a rapidly growing attention over the last few years due to its salient properties.

Mobile Agent (MA) is a software abstraction (program) that exhibits features of ***autonomy***, ***social ability***, ***learning***, and most importantly, ***mobility***. MAs migrate from one machine to another across a heterogeneous network (hence mobility), at times and to places of their own choice (hence autonomy).

They transport their state from one environment to another, with their data intact, and still being able to perform appropriately in the new environment. When an MA decides to move, it saves its own state and transports this saved state to next host and resume execution from the saved state.

According to (de Groot, et al. 2005) software agents themselves are assumed to have the following properties:

a) ***Autonomy***: they have control over their own actions and state.

b) ***Communicative (Social ability)***: they can communicate with other agents.

c) ***Reactivity***: they react to changes in their environment.

d) ***Pro-activeness***: they make plans to reach their goals and can take initiative to pursue these.

Additionally, agents are assumed to be intelligent (they can reason, learn and adapt) and mobile (they can move between network-connected computers).

(Katsumo, Murata and Tokro 2007) stated the importance for MA systems to support the four main properties: *Migration*, *Communication*, *Access Control of Variables*, and *Reference to Remote Variables*.

Migration: MAs are required to migrate from one site to the other in a networked environment. When they move between hosts, it is necessary to preserve their state.

Communication: An MA needs to communicate with other agents in order to access databases or exchange information.

Access Control of Variables: It is important to protect resources an MA might carry, and this may require specifying access control for certain variables.

Reference to Remote Variables: To minimize the overhead of migration in case of large amount of data, it may be suitable for the agents to be able to reference such data remotely.

3.2 Benefits and Limitations of Mobile Agents

Though various important points are stated by different researchers in a distributed system, MAs are often dispute areas that programmers and system developers often argue about their benefits over their limitations. In this subsection we will summarize the benefits and limitations of MAs.

3.2.1 Benefits of Mobile Agents

From the computational aspect and resource utilization on networked environment MAs can provide lots of advantages some of which are:

a. ***Improved locality of reference***: The motto in MAs is: "*move the computations to the data rather than the data to the computations*". The principle is that local method call is much faster than remote procedure

call such as CORBA, RMI, and DCOM. It also reduces the load on the network since there will be no large data movement.

b. ***Ability to represent disconnected users***: MA architectures can solve the problems caused by intermittent or unreliable network connections. Since MAs can execute asynchronously and autonomously, they can be dispatched into the network where they work independently and finally return results when the host reconnects to the network.

c. ***Flexibility and Scalability***: Flexibility and scalability allows agents to play a role in the development of active and dynamically managed networks and distributed systems, hence a system with MAs provides advantages such as:

- ***Space savings***: MA code and state do not need permanent storage on the hosts they run on.
- ***Reduction in network traffic***: MAs are based on the concept of bringing computation to data rather than data to computation.
- ***Asynchronous autonomous interaction***: A MA can be delegated to perform a certain task and the agent is intelligent enough to decide, at execution time, with whom it needs to communicate.
- ***Interaction with real-time systems***: MAs can be installed close to real-time systems to prevent delays caused by the network congestion.
- ***Robustness and fault tolerance***: MAs can be used to increase availability of certain services by assigning individual agents for each service. The system will have better tolerance to network faults – able to operate without an active connection between hosts (clients and servers).
- ***Support for heterogeneous environments***: Mobility frameworks provide the required support for MAs, and thus agents are separated from the host and its operating system. It also allows asynchronous execution on multiple heterogeneous network hosts.
- ***Online extensibility of services***: MAs can be designed to extend capabilities of applications. The system has capability of dynamic

allocation – actions that are dependents on the state of the host environment.

d. ***Customization***: MAs are customized for end-users (clients) to carry out specific operations. They are also flexible for maintenance – to change an agent's action; only the source rather than the computation host must be updated with the change.

3.2.2 Limitation of Mobile Agents

However, MAs are having the benefits listed above, they do not offer this advantageous for free. They are having their own limitations and drawbacks such as *complexity* and *security*.

a. ***Complexity***: Mobile code – based systems seem to be quite complex both to design and to maintain because of their development and standardization. They lack integration and interoperability with legacy systems, and the design and standardization of feasible infrastructures for their basic services to carry out.

b. ***Security***: Security issues, such as protecting the agent from being abused by malicious hosts and protecting the host from being attacked by malicious agents, have to be resolved.

Due to their mobility nature, MA-based systems are vulnerable for security threat. Protection of agents from malicious access while they are in transit between two authenticated places can be achieved using standard secured communication mechanisms such as secure socket layer (SSL).

The other major threats of MA-based systems are: attacks by platforms and attacks by agents.

i. **Attacks by platforms** may be made on to agents that are migrating to the server in order to steal the information carried by them or by mimicking itself as trusted site and generating agents that may migrate to other trusted platform in negative aspect. The problem of protecting the agent from malicious places is caused by the fact that the code of the agent is executed in an un-trusted environment.

ii. **Attacks by the agents** may happen to both trusted platforms and agents. Other agents that are deployed from un-trusted sites can also be a treat to the platforms listening for agents or even to other legitimate agents in the fly.

3.3 Mobile Agent-based System

A MA-based system relies on the advantages of MAs handling the drawbacks associated with them. Many MA solutions cannot be adopted because of security threats. Hosts and networks must be protected from malicious agents; agents must be protected from other agents; and agents must be protected from malicious hosts (platforms). The two possible treats by the agents on a platform and other agents can be solved by: ***Authentication***: which aims to make sure a client (or agent) is who he claims to be, and ***Authorization***: that grants access to resources to an authenticated client (agent).

3.3.1 Trustworthy Mobile Agents

Designing a mobile system that addresses the security issues can then be achieved through trustworthy MAs with the use of authentication and authorizations. Often trustworthy agents are built in unrealistic infrastructure. However, building trustworthy agents that perform their missions with confidence even though they sometimes execute on un-trusted hosts requires several criterion to fulfill, such as:

High Mobility: Agents must be free to migrate to, and execute on, a wide variety of hosts those are unknown to the users who launched the agents. Without such mobility, agents will be unable to perform the searching and commercial operations often envisioned for agent technology.

Detached Operation: Agents must operate autonomously, without the need for constant communication with users, and, preferably, without constant communication with trusted infrastructure elements which may or may not exist.

Extended Deployment Periods: Agents must function for extended periods of time, thus allowing users to launch long-term "watcher" agents that take action only if specified criteria have been met and other long-term service agents.

Safe Execution: Agents must be free from integrity attacks conducted by malicious hosts or other agents, and must be protected from faulty execution or non-execution by malicious hosts. Agents will also be much more useful if they can carry secrets (such as cryptographic keys or user decision information, such as how much a user would be willing to pay for merchandise).

Realistic Infrastructure Requirements: Agent properties must not rely on unrealistic infrastructure assumptions, such as the assumption that all hosts are trustworthy, that implementation algorithms will remain secret, or that every agent execution environment is implemented by a tamper-resistant hardware peripheral.

In addition to the use of trustworthy MAs to address the security treats various methods have been proposed for protecting the agent platform and the agents as well. Some of these are summarized by (Jansen and Karygiannis October 1999) in to two classifications as countermeasures for ***protecting the agent platform*** and ***protecting agents***.

Techniques devised for protecting the agent platform include: *Software-Based Fault Isolation*, *Safe Code Interpretation*, *Signed Code*, *Authorization and Attribute Certificates*, *State Appraisal*, *Path Histories*, and *Proof Carrying Code*. On the other hand, some of general-purpose techniques for protecting an agent include: *Partial Result Encapsulation*, *Mutual Itinerary Recording*, *Itinerary Recording with Replication and Voting*, *Execution Tracing*, *Environmental Key Generation*, *Computing with Encrypted Functions*, and *Obfuscated Code* (Time Limited Black-box hiding the agents).

3.3.2 Activities of Mobile Agents

During their life time MAs migrate to different sites or places, P_i, for the execution of their tasks at various stages, S_i. At every site the agent will be having potentially multiple set of actions (operations), a_i, to be completed. Between the source (a place where the agent is beginning its stage or where it is created) to the place where it will perform its final execution stage (known as destination), an agent undergoes through various intermediate stages possibly on different places. These activities of MAs are described in (Pleisch and Schiper 2004) with Figure 3-1.

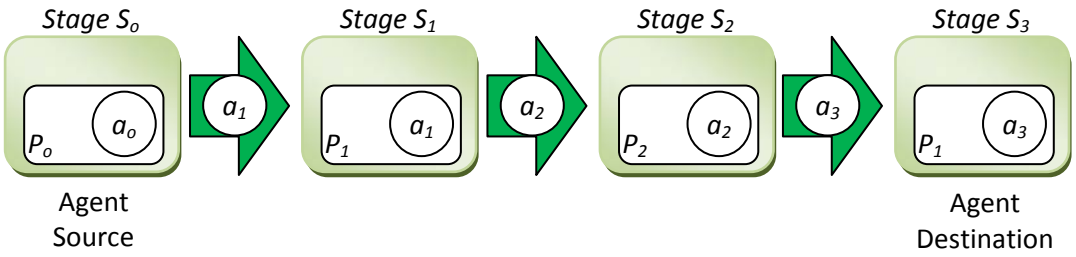


Figure 3-1 Execution model of mobile agents (Pleisch and Schiper 2004)

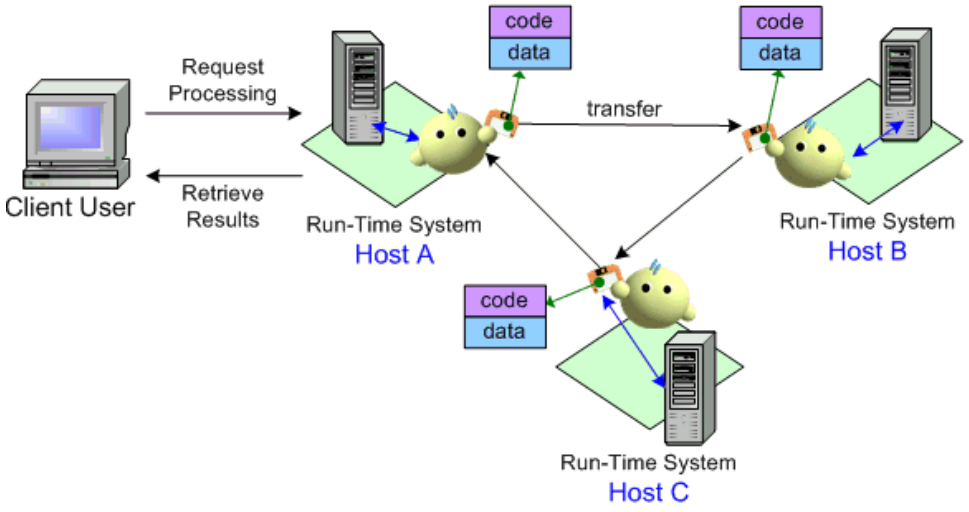
The detail history of the trajectory (or the list of sites that are visited by the agent from its source place to its destination P_0, P_1, $\cdots$ P_n) is known as ***itinerary***. The itinerary of the MAs can be used in two options to maintain a secured system. One possible option is the user launching the agents will list the itinerary containing the servers that are trusted to the system and the sequence in which the agent needs to dispatch itself. This will allow the agents to work only in a controlled environment that protects them from malicious platform attack to the agents. The Second option protects agent platforms (servers) from malicious agents attack by writing the history of the agents dispatched places in the itinerary. A server accepting a request from upcoming agent will detect the history of the agent and will inspect a treat from the trajectory of the agent has been following which may include non-trusted sites.

In order to perform their tasks, MAs are composed of:

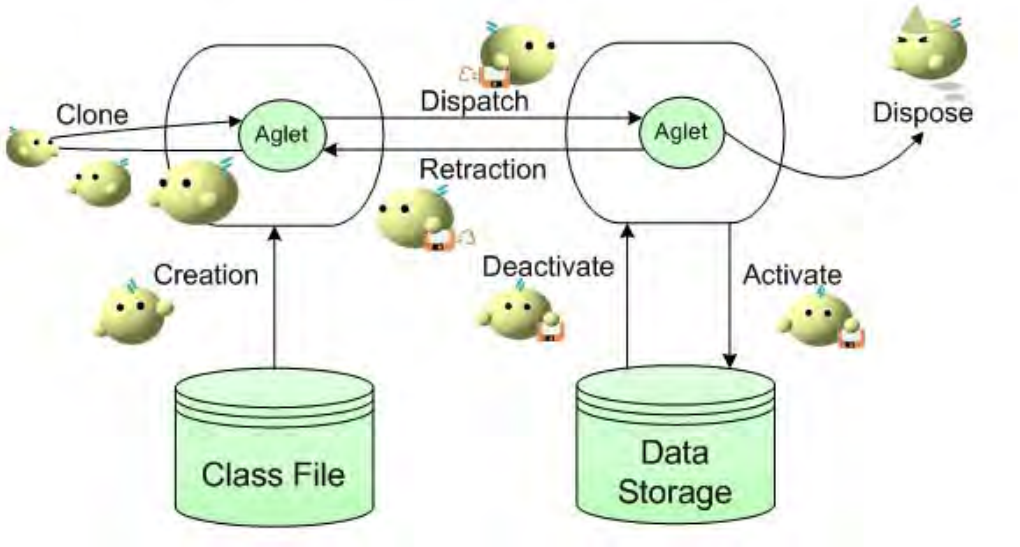
a. **Agent code**: a code that is given as a task to be performed by the agents in its life time. The agent code may consist of various stages, S_i, that need to be executed possibly at different sites.
b. **Agent execution thread along with an execution stack**: in order to maintain their various stages and coordinate their task, agents will be having execution threads and stack that holds current stage or state locator. The thread will be helpful to log a report of the activities of the agent for its stages.
c. **Agent data part**: the data that is required by the agent for its execution. The weight of the data handled by a single agent need to be as light as possible to utilize network resource.

The diagram from Mobile Computing Laboratory, Department of Information Engineering and Computer Science, Feng Chia University (Wikipedia, the free

encyclopedia - Mobile Agent 2006), Figure 3-2 indicates the concepts and activities of MAs.



(a)



(b)

Figure 3-2: Concepts and activities of mobile agents

As described in the figure, MAs undergo through various states in their life time starting from their initiating site (source site) to their destination. The major activities or states as depicted in the Figure 3-2(b) include:

Creation: the agents are created at their source site to be given specific site they need to report to, the initiating site or other delegated site.

Activated: created agents at the source will be given their task and accordingly they act upon to perform their duty at the source or any other site that the agents need to migrate for completing their duty.

Deactivated: agents as being mobile they are going to go through various sites in doing so their task may be suspended during transition or waiting for resource. The agents that are migrating to other sites or waiting for resource are entering to a temporary deactivated state. Once the migration process is completed or the resource is secured, the agents will return to their execution of task that leads them to the activated state.

Dispatched: migration of an agent into a favorable environment required by the task assigned leads the agent to go through a dispatched state. This state is occurring from the moment an agent is preparing itself and establishing connection to the migrating site until it is transporting and reconstructing itself successfully to the migrated site.

Retracted: in case of successful completion or unsuccessful attempt of task assigned to agents at migrated sites, agents may need to report back to the site creating them or other delegated site. In such cases, an agent needs to retract and come to the site where it is called upon to report.

Disposed: agents that are completing their task or no longer needed by the system will be disposed and release all the resources they are utilizing.

Another class of MAs' evolution state considered in (Deng, et al. 2002) has four states of MAs:

Ministering: in this state the agent is actively serving the user,

Suspending: a state of an agent waiting for resource or connection in the environment that forces the agent to suspend its task,

Dangling: a state of an agent that has lost connection to the server that is waiting for readmission or a one that may be disposed, and

Mutating: a state of an agent that has changed its identity and readmitted to the system.

The researchers presented the transition or flow of the agents' evolution state as shown in Figure 3-3. The agents born or created by the user are admitted to the system and given a state of *ministering*.

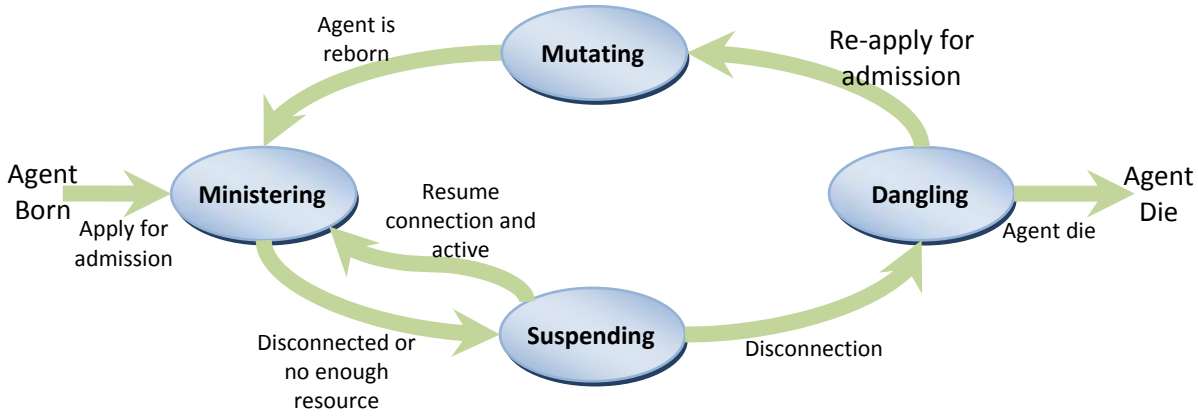


Figure 3-3: Mobile agents' evolution states transition diagram

Agents in the state of *ministering* may be entering to the *suspending* state when they are waiting for resource or preparing themselves for migrating. *Suspended* agents may continue their execution after securing the required resource or locating themselves to the destination node. In case of communication lose, the agents may undergo into the *dangling* state. Agents that have lost connection to the server may die out or they may go into *mutating* state to enter to the system again and continue execution at the state of ministering.

3.3.3 Mobile Agent Applications

Though, they are having high threat of security still MAs are powerful tools for implementing distributed applications in networked system. As presented in (Milojicic, Dejan; Johansen, Dag; Kotz, Dave; Lange, Danny; Petrie, Charles; Rygaard, Chris; 1999), MAs are helpful in designing of a system.

- Where agents are launched by an appliance, and/or
- Where a user can ship and install agents representing it more permanently on a remote server.

In (Milojicic, Dejan; Johansen, Dag; Kotz, Dave; Lange, Danny; Petrie, Charles; Rygaard, Chris; 1999), it is pointed out that the most common application domains (areas) for MAs include:

i. **Disconnected computing (Close interaction)**: in this category of applications the agents are to be launched by the system to the favorable environment, where data is available. The requesting client is then free of its will to lose connection to the server which will report back to the client only when needed. Some of the applications in this category may include: matrix computation and statistical analysis.

ii. **Data intensive applications**: applications that require remotely stored data to be processed. In such applications the agents are migrating with the processing code to the location of the data (server containing the data).

iii. **Electronic commerce**: in the current advancement of networking there are millions of sites that are having their products available for sell on the internet. For areas where an individual needs to search number of sites for items MAs will play a major role in representing the user at the sites. The agents can be deployed across the sites and search for the items and report back to the user which will help to reduce the network traffic and the time required by the user searching every available site.

iv. **Dynamic deployment of software**: in such applications the user is shipping agents to a site where the deployment of the software is required to act on its behalf. A typical example in this is; system administration and management, an application where the agents can represent system administration tasks.

v. **Information retrieval**: is also another application area where MAs can be of great help in managing, discovering and monitoring resource availability.

vi. **Memory constrained environment**: are application areas as in the embedded system that are having limited resource. In such areas, MAs can be used to process the request by the resource constrained system in a better environment for computation and return back to the user with the result.

vii. **Network management**: applications that make use of MAs can also be advantageous in network management systems to watch out clients in different aspects such as security, resource utilization and so on.

3.4 Approaches for Multi-Level Transaction using Mobile Agents

Handling MLTs in different applications can be implemented with the use of MAs that will be delegated for the transaction at various levels. The four most common MA-based approaches to the MLT management are: *Centralized platforms – centralized agents*, *Decentralized platforms – centralized platform's agents*, *Centralized platforms – decentralized agents*, and *Decentralized platforms – decentralized agents* architectures as presented in (Terziyan and Khriyenko 2004).

Centralized platforms – centralized agents: Figure 3-4 shows an architecture where each platform in the system registers its services at some central platform of the network. The platform gets direct requests for services from clients and decides to which platform to forward the request. When a local platform gets a forwarded request, it analyzes the request and decides to which agent (service) on the platform to forward it.

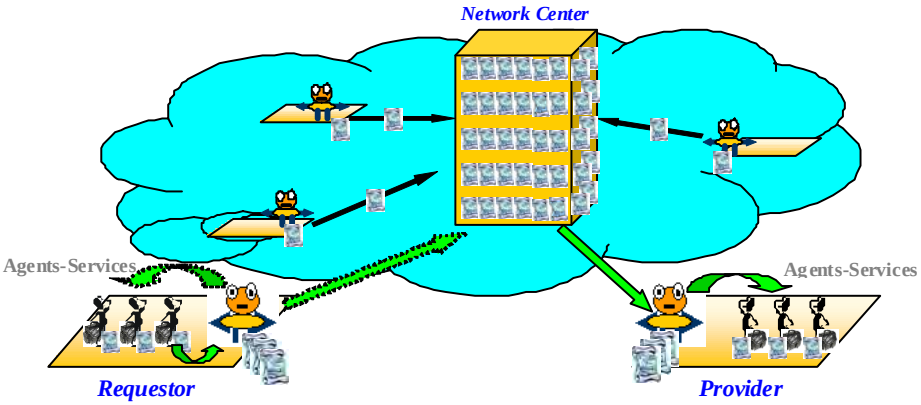


Figure 3-4: Centralized platforms – centralized agents approach for MLT

Decentralized platforms – centralized platform's agents: Interoperation between platforms is based on a P2P semantic service discovery. In such architecture shown in Figure 3-5, the nature of the servers allows the use of a P2P decentralized distributed network.

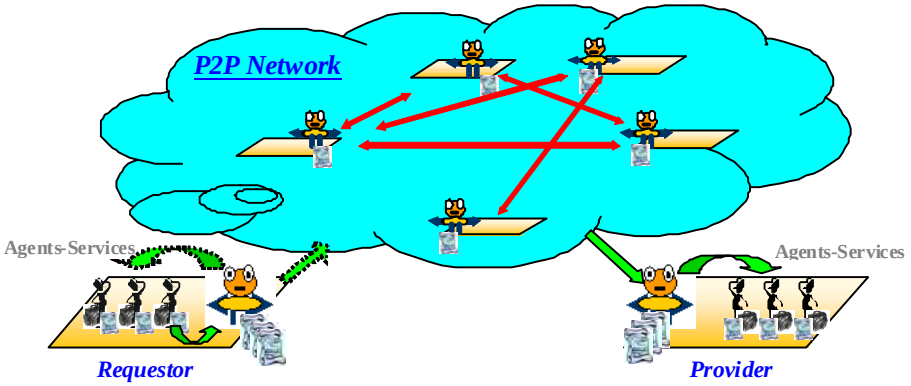


Figure 3-5: Decentralized platforms – centralized agents approach for MLT

Centralized platforms – decentralized agents: In Figure 3-6 the central point of the network selects the platform, which is assumed to be able to serve the request. Inside the platform, which finally gets the request, the right servant will be found based on a P2P semantic search within the platform.

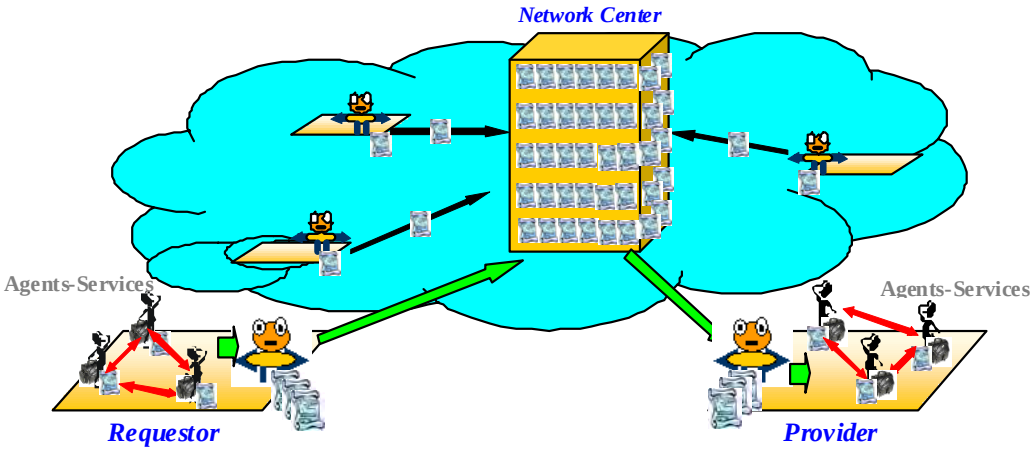


Figure 3-6: Centralized platforms – decentralized agents approach for MLT

Decentralized platforms – decentralized agents: Both interactions within the platforms' network as a whole, and locally within each platform follow the P2P architecture. Such applications will improve robustness as it eliminates a problem of single failure that can force the whole system to be down at the time of one site failure. When one of the sites is down the system can still run from the remaining sites as they may take over the task given to the failed site. Figure 3-7 shows the diagrammatic schema of the decentralized platforms – decentralized agents approach for MLT of agents.

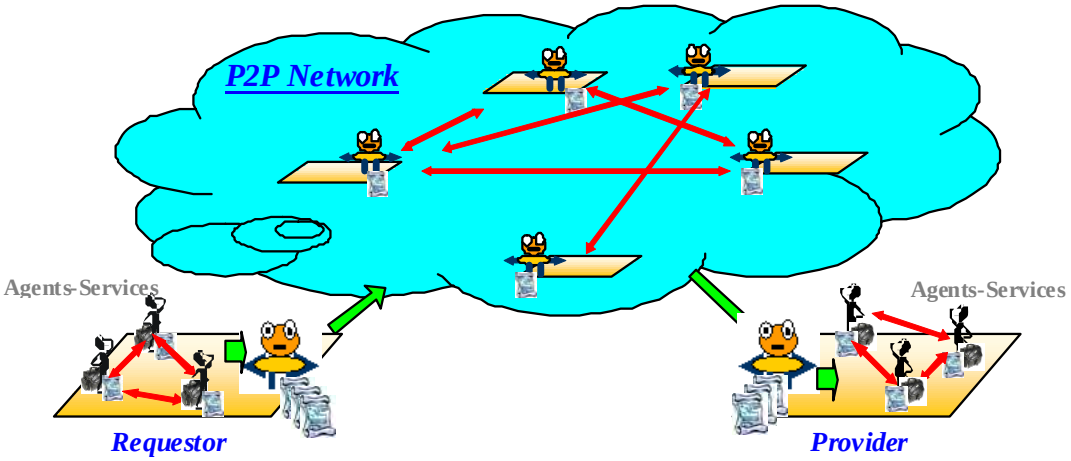


Figure 3-7: Decentralized platforms – decentralized agents approach for MLT

3.5 Related Works with Mobile Agents

From the advent of the inter-networking system there has been number of researches going on to improve distributed system applications. The researchers in (Eid, et al. 2005) tried to pin point the application areas and trends for MAs. In their study the mobile applications they reviewed on multimedia were consuming 10% of the overall various classes and 13% were dealing with information searching and filtering, and they found that:

a) MAs have the potential of becoming general framework in which a wide variety of distributed information search and retrieval applications can be implemented efficiently, easily, and robustly.

b) QoS related issues have received the lion's share of the agent-related work in the domain of multimedia, which is not surprising knowing that the delivery of multimedia content across computer networks must adhere to certain criteria, most important of which is minimizing delivery delays. In fact the main reason why MAs were applied is due to their potential for minimizing network traffic and speeding up delivery.

c) An open and evolutionary path that requires the effort of the research community is how to extend the current MA-based systems to support accounting functionalities.

3.5.1 Mobile Agent-Based Applications

In the area of MAs for distributed systems tremendous contributions have been made in proposing different applications. In this sub section we will discuss some of the works that have been tested and implemented with the use of MAs.

Applications in Resource/Data Management

(Trappey, et al. 2004) has presented an MA-based system for distribution and logistics management. The system involves many participants such as freighters, distribution centers, manufactures and distributors that are represented with MAs. The main purpose of the researchers was to provide an agent-based approach to add agility to the comprehensive logistics service tracking across organization boundaries.

The proposed architecture by the authors is further divided into tracks for the dealer side and the data side of the logistic service. The agents on the dealer side are responsible for coordinating the entire agent system while the agents on the data side are responsible for the data collection from local database. Receiver, Agent assignment and dispatching, Coordinating agent and Presenter modules are included on the dealer side. And the data side consists of three modules: Coordinating agent, Information collector (grabber) and Presenter.

In (Deng, et al. 2002), the researchers proposed a system that allows mobile persons (students, instructors and system administrators) to establish communications through an agent communication framework. They say that for a roaming nature of the objects in distance learning MAs are best fit to provide a persistent environment that allows them to obtain their personal profiles. In this regard, the mobile person agent serves as the front end of the system proposed.

The researchers pointed out that, OO approaches are the role settings in the design of MA-based architectures. They also suggested that a framework containing of MA communication network will aid software developers with agent computing routine management task, an approach that will add flexibility and scalability in the user's utility tool.

The use of MAs on the synthetic aperture radar atlas (SARA) digital library (Yang, et al. 2000) has elaborated the potential of MAs in the application of distributed heterogeneous remote sites. The authors presented an architectural

design and test implementation that enables automatic and dynamic configuration of distributed parallel computing and efficiently supports on-demand processing with the use of MAs for remote sensing data about the earth environment. The MA-based architecture of the SARA system is shown in the Figure 3-8.

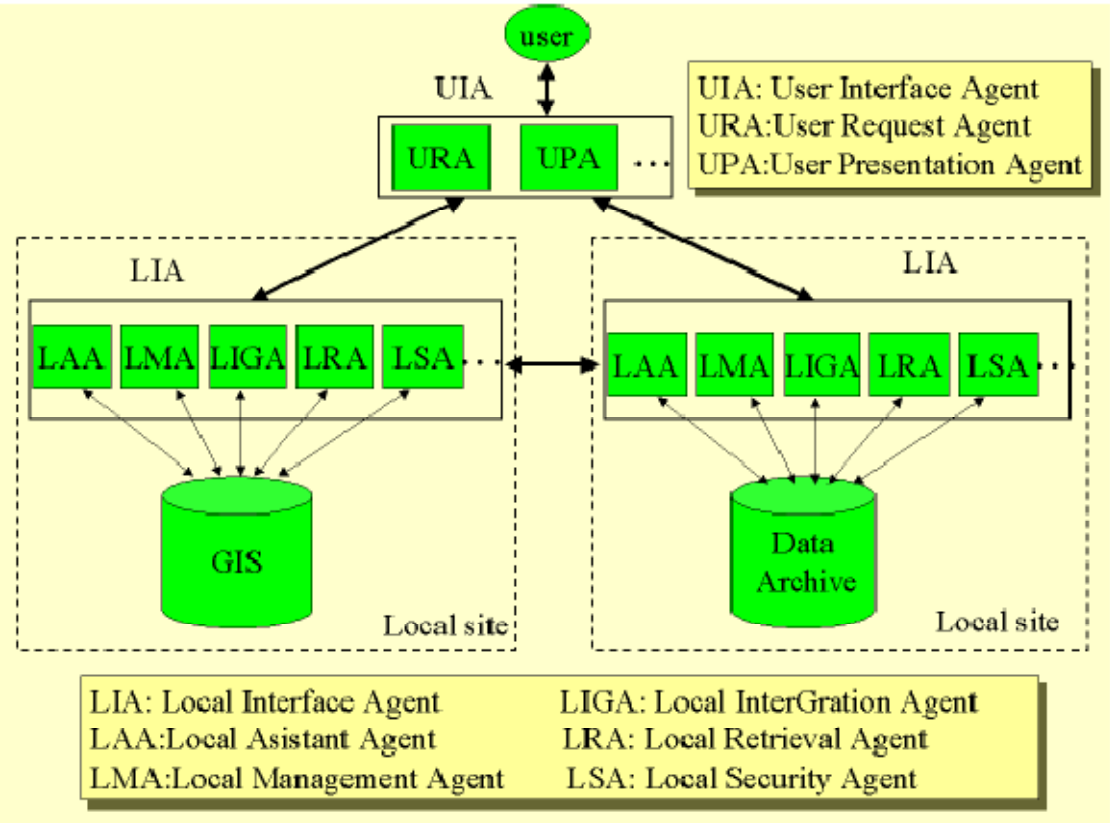


Figure 3-8: Mobile agent-based architecture of (Yang, et al. 2000)

The agents in these authors work are grouped into two as:

i. ***User interface agents***: these are front end to the user. These are of two kinds: User request agent and User presentation agent that handle the user request and presentation, respectively.

ii. ***Local interface agents***: providing an extensible set of services such as: Local assistant agent, Local management agent, Local integration agent, Local retrieval agent and Local security agent that are responsible for the services their naming implies.

The work of the authors in the same presented the fact that MA-based architectures are far more suitable for applications that require distributed access, concurrent querying and parallel computing over multiple and heterogeneous

remote sensing archives in a modular and scalable fashion. Moreover they have also proven that the principles of MAs can be applied on automatic and dynamic configuration of distributed parallel computing.

On the other hand, the author in (Moser 1999) with Prithviraj Dasgupta and Nitya Narasimhan developed and implemented an MA-based electronic trading system with the use the of the Java MA application aglet. The system named MAs for networked electronic trading (MAgNET) launches MAs in search of items by the users in a networked system. The system developed by the authors represents buyers through MAs that are given itinerary (address of suppliers) to visit. The MAs in the system then return to the buyer after visiting the supplier in the itinerary list with a set of quotation for the item. The buyer then may decide to confirm or cancel reservations made by the agents for attractive offers. If confirmed the agent will communicate to the surrogate agent left behind at the time of visit at the supplier presenting the attractive offer, and the transaction will take place.

The security threat due to mobility of agents in the electronic trading system was handled by the author with the use of secure server protocols at the supplier side. The author claims to solve the problem of security regarding threats to the MAs from malicious supplier sites by the use of once-only usable lock mechanism that requires authentication from the origin site of the agents.

The objective of the research work in (Moser 1999) was only to address the problem of optimal point of exchange between buyers and suppliers that can be extended from suppliers to part suppliers creating a chain of trade. The work similar to the other related researches also presented the potential of MAs in a distributed networked system.

The research work in (Dale and DeRoure 1997), on the other hand has presented an architecture where MAs move across distributed information management (DIM) environment managing distributed data. There were three basic DIM agents that were created in their implementation:

i. ***Resource discovery agent***: that will be responsible for search of information or document within the domain of the system,

ii. ***Link assertion agent***: that identifies valid and invalid destinations given the hypermedia links, and

iii. ***Link update agent***: that updates changes in hypermedia links in the DIM.

There were also other agents to present the services such as FTP resource agent, WWW resource agent and Domain agent. The interactions of the agents in this research work were presented as in Figure 3-9.

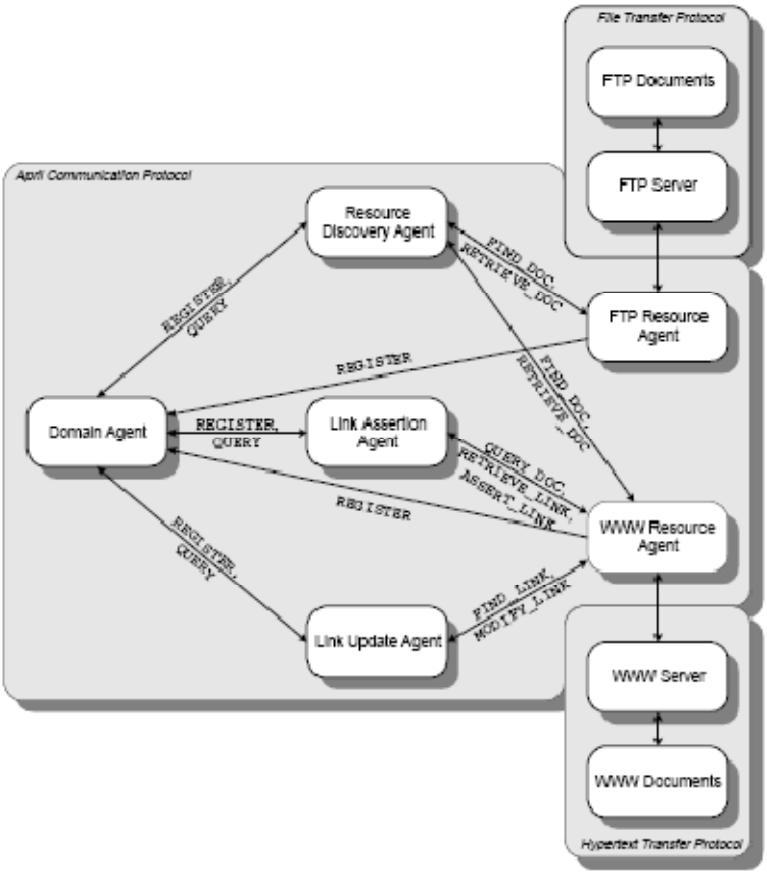


Figure 3-9: DIM agents' interaction of (Dale and DeRoure 1997)

The approach taken by the authors was OO for both primitives and agents in their design. However, the system developed in their work was targeting specific areas where Internet-based tools are lacking or not available.

Applications in Multimedia

The researchers, (Manvi and Venkataram 2006) in their paper "Agent-Based Subsystem for Multimedia Communications" presented issues in multimedia communication as resource allocation, QoS routing, synchronization and play-out compensation. In their paper, they proposed a subsystem named Protocol

Engineering and Technology unit – Agent-based Subsystem for MultimediA Communication (PET-ASMAC) that is simulated in several network scenarios.

According to the authors some of the QoS parameters in a multimedia communications are:

- **Bandwidth**: the data transmission rate measured as the amount of data to be transferred for every second. For multimedia applications several Mbps (Mega bits per second) bandwidth is required, otherwise the application will be delayed or their will be loss of data.
- **Transfer or end-to-end delay**: the time between the multimedia data retrieval (or generation) at the source and presentation at the destination.
- **Jitter**: the variance of the transfer delay that need to be minimized to create the impression of continuous play in a video presentation.
- **Loss rate**: that measures the reliability of the network media. The loss rate has to be minimized to extent of being zero to boost the reliability of the system.
- **Synchronization**: that expresses the temporal relationship between streams and the presentation units of a stream.

The system proposed by the authors addresses the multimedia communication requirements using MAs that is more robust and has a smooth and continuous presentation in a point-to-point communication. The system is claimed to be flexible, adaptable and support component-based software engineering. The authors' objective in (Manvi and Venkataram 2006) was to improve the QoS requirements of a multimedia application by:

- Finding a path which satisfies the QoS requirements,
- Negotiating/renegotiating the resources,
- Monitoring data rates, delays, change of delays and losses,
- Computing the play-out timings,
- Changing the transmission and play-out rates, and
- Deploying different types of synchronization.

Their system uses three types of agents: *communication manager agent*, *QoS negotiator/renegotiator agent* and *delay estimate agent* as indicated in their component diagram Figure 3-10.

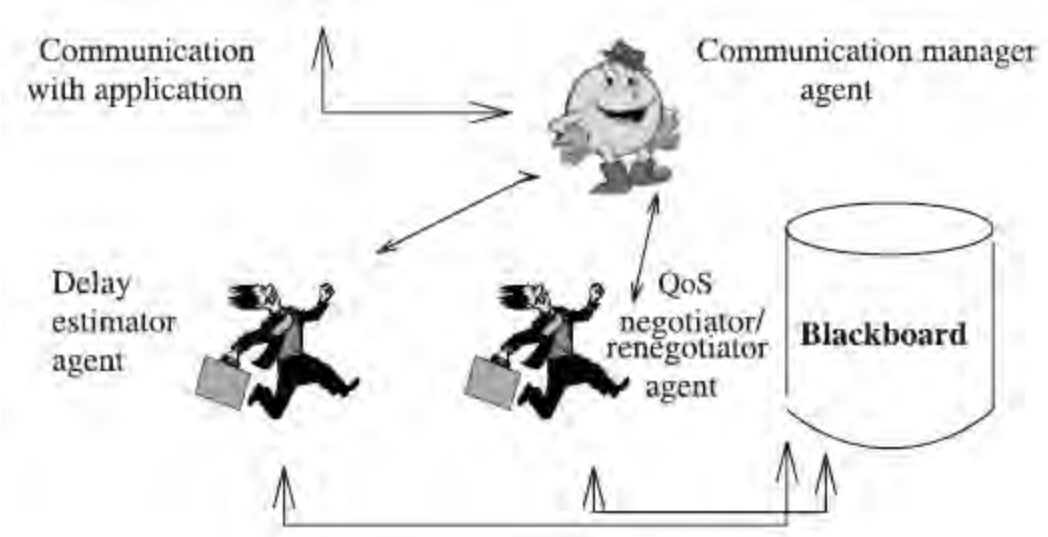


Figure 3-10: Components of PET-ASMAC in (Manvi and Venkataram 2006)

i. ***Communication manager agent***: a static agent that manages the other two types of agents and monitors QoS parameters of an application stream at the network nodes.

ii. ***QoS negotiator/renegotiator agent***: an MA that is used to find a QoS route satisfying bandwidth, delay and loss requirements of an application.

iii. ***Delay estimate agent***: an MA that is used to detect the clock time difference between the server and the client.

They found out that the use of MAs for the PET-ASMAC multimedia communication offers flexibility, scalability, efficiency (efficient resource allocation), customization, adaptability, software reusability and maintainability.

(Satoh 2000) also presented a framework for constructing networked applications, including multimedia ones. The framework is based on a hierarchical MA system for allowing more than one MA to be dynamically assembled into a single MA. It is built on the Java virtual machine and MAs are given as Java objects

To demonstrate the utility of the framework in the construction of networked multimedia applications the researcher built two sample examples: MA-based e-

mail system and video-on-demand system. The video-on-demand system consists of a server and its clients. The server is taken as a file-sharing server for allowing multimedia contents to be stored and transmitted; and the client receives the transmitted multimedia contents from the server and supports user interfaces for presentation.

Other researcher in (Arora, et al. 1999) also, built a prototype with the use of MAs that travel across a wide area network to analyze documents for the content search at the location of the source. The system allows different pattern recognition algorithms to be applied for the retrieval of images. The algorithms are encapsulated to a set of MAs. The approach allows reducing the network traffic and computation problem. It uses five classes of agents:

i. ***Media search agent***: that is mobile and uses recognition algorithm in identification of color or simple shape,

ii. ***Repository manager agent***: that encapsulates details of repository and presents a standard view for an image collection,

iii. ***User interface agent***: an interface that provides a tool to build a query of content base image retrieval to the system,

iv. ***Search coordinator agent***: that manages a team of agents required for retrieval of image and coordinates their activities for optimal retrieval, and

v. ***Registration agent***: an agent that registers the agents in the system.

The system is implemented with the use of Java RMI for the network communication. It comprises of seven search agents of which five are generic that implement standard algorithm of shape or color recognition. It is tested only for image type of multimedia objects and it basis the problem for content-based search of stored image objects.

The authors claimed that the agents can be dynamically selected for every retrieval problem and their logical structural composition allows the search agents to operate over heterogeneous repositories. The registry system provides a mechanism for selection of agents and appropriate features that are necessary for the retrieval.

3.5.2 Agent Systems in Java

Most of the prior works that designed MA-based applications had used Java programming. Java provides a secure execution environment, enabling maximized flexibility in using agents without worrying about the possibility of a hostile agent destroying the local host. Being an interpreted language, it is able to execute the same set of codes on multiple platforms. In this aspect that most mobile applications are tested and employed in a Java environment.

Some of the MA systems that are developed in Java are Aglets, Java–To–Go and Agent Programming System and Language.

Aglet

Aglet is a Java MA platform and library that eases the development of agent-based applications. An aglet is then a Java agent able to autonomously and spontaneously move from one host to another. It supports asynchronous messaging in communications between agents. The migration of agents is implemented with the use of Java Object Serialization library.

Java–To–Go

Java-To-Go first developed by (Li and Messerschmitt December 1996) is a set of Java-based interface and class definitions that implements a distributed cross-platform environment for task-oriented MA experimentation. Within "Java-To-Go", MAs are implemented as threads and conducted inter-agent exchanges. Java–To–Go supports multicasting messages to a group of agents in communications. It also implements migration of agents using Java Object Serialization library.

Both Aglet and Java–To–Go do not support state preservation of MAs since local variables are not maintained in Java Object Serialization library.

Agent Programming System and Language (APSL)

Agent Programming System and Language (APSL) developed by (Katsumo, Murata and Tokro 2007) is an MA system composed of three parts: *Mobile Agent*, *Place Agent* and *Communication Agent*. The mobility feature is supported by the MAs that move around host and communicate with the place agents. The place agent that resides on the service host provides variety of services for the MAs.

The communication agent; on the other hand, supports communication between the MAs and the place agents.

APSL attempts to overcome the use of the Object Serialization library that has a problem in local variables that are not maintained, and execution of progress of objects that could not be maintained thereby, preventing migrated agents from resuming their executions.

In order to avoid these problems, APSL does the following:

a) Its compiler translates all local variables in the main method to instance variables, and

b) Each MA maintains a variable holding the current execution step number to jump to the next intended execution step after resuming.

There are as well some other agent programming systems which are not based on Java such as: Telescript and Agent-Tcl. Telescript is well-known as it is the first agent programming system. It preserves the agent's state when it migrates so the execution can be resumed following migration onto the destination host. Agent-Tcl addresses the mechanism of disconnection and reconnection of the client's host.

As can be seen from the works done so far on MAs, they are having number of applications in the current distributed system. Also from the fact that the use of multimedia objects is increasing from day to day, MAs have shown a considerable share contributing for the improvement of QoS in multimedia supported systems.

The researchers in the review have all tried to point the applicability of MA architectures in various distributed systems. The application areas in which the MA architectures are tested differ widely, and researchers implemented different types of agents that are associated with the basic nature of MAs. Though the applications tested are wide in variety, all are alike in their basic nature of being distributed system management.

All the research works preferred that MAs to be implemented in object–oriented approach to increase the architectural expandability for new modules and transactions. The approach allows the system to be flexible, scalable,

customizable, adaptable, and efficient in resource allocation and utilization. It also improves the software reusability and maintainability.

In general, the QoS for MAs systems are found to be bandwidth utilization, security and efficient data presentation and management representing the mobility nature of the problem. In multimedia applications temporal constraints are also included in the QoS of the system. Moreover, in database applications MA base architecture can also be designed to support the required middleware for the transaction management.

To the knowledge of this researcher none of the works done so far are attempting to address the MLT of a DMMDB through MAs. Though, there are many researches in the area they are not tested for MLT. In this regard, this thesis work studies MAs for MLT management of MMDBs. It proposes an MA-based architecture for managing the MLTs of a DMMDBS for decomposition/composition of multimedia data transactions.

4 Architecture for the Multi-Level Transaction

The literature survey in chapters 2 and 4 presented the need for a middleware that handles the transactions of MMDB applications. It is noted that the middleware is required to be a lightweight system that supports the QoS for presenting the multimedia data in efficient resource utilization such as bandwidth and disk storage.

An MA based architecture in DMMDB is studied and designed to allow distributed access and concurrent querying over multiple and heterogeneous multimedia systems for MLT in a modular and scalable fashion. In this chapter, we will discuss the proposed MA based architecture for the middleware.

4.1 Multi-Level Transaction of Multimedia Database

Though there are number of issues on distributed multimedia applications as stated in section 2.3.2, such as MLT, media synchronization, content-based searching and so on; the issues studied under this thesis work focuses on the MLT for:

a) **Distributing and Storing**: As the nature of media object is large and resource consuming storing media files in a single platform will have drawbacks in storage as well requiring large data movement in retrieval from other distributed sites. Instead, distributing the media files across distributed sites may reduce the required storage at one node as well provide an option to bring data closer to the requiring site on storage.

b) **Indexing, Querying and Presentation**: The key functionality in an MMDB is how to access and how to exchange multimedia information effectively, this includes:

   - **Indexing**: indexing of stored data is the best method in database systems for efficient retrieval of information,

- **Querying**: a media application requires best querying feature for media object retrieval, and
- **Presentation**: the media file located on the distributed database required to be presented in reasonable response time and less resource utilization.

In the process of distributed multimedia management the two most important operations are ***decomposition*** and ***composition*** of the media file. During distributed storage, the media file need to be decomposed into parts that will be either fragmented or replicated over the nodes in the distributed system. If the multimedia data type is required for searching and indexing such as title, format and descriptions of the media file, the data may be replicated over the sites to provide faster and efficient operations upon media retrieval. However, if the data is the decomposed part of the media file that consumes data storage such as video frames, after decomposition it may not be wise to replicate it over every node instead the data is fragmented over the multiple nodes.

The decomposition/composition procedures applied in this research work are decomposing the media file from its byte level. Hence, group of bytes in the media forming frames extracted and the set of frames form slices of the large media files. Threaded tasks are used to process the decomposition/composition of data upon media file uploading (saving) and download streaming (retrieval).

These tasks are performed in an MLT consisting of three layers as shown in Figure 4-1.

Figure 4-1: The three layer multi-level transaction

- **Index layer**: an object layer in the middleware of the proposed architecture that handles all the indexing of media files,
- **Data layer**: a page layer managed by a standard relational DBMS for storage of the multimedia data, and

- **Presentation layer**: a layer acting as a user interface to accept uploading files and streaming upon retrieval.

The transactions in the three layers are processed in parallel with threads run by the agents in the system to represent the requirement of concurrency. The concurrent transactions are executed possibly on distributed sites.

4.2 The Agents' System Architecture

The proposed agents' system architecture (ASA) consists of agents running under the agent platform (AP) that runs on every site of the distributed system. The AP is a middleware that all running applications contact to perform their MMDB transactions. The transactions considered are the ones that are used in uploading a media file and downloading and streaming a media file which are mainly insertion of data and retrieval in response to the user searching criteria. The proposed architecture of the system as shown in Figure 4-2, is designed in the P2P network discussed in section 2.3.3. Each node is equal importance in the distributed system equipped with the AP and DBMS.

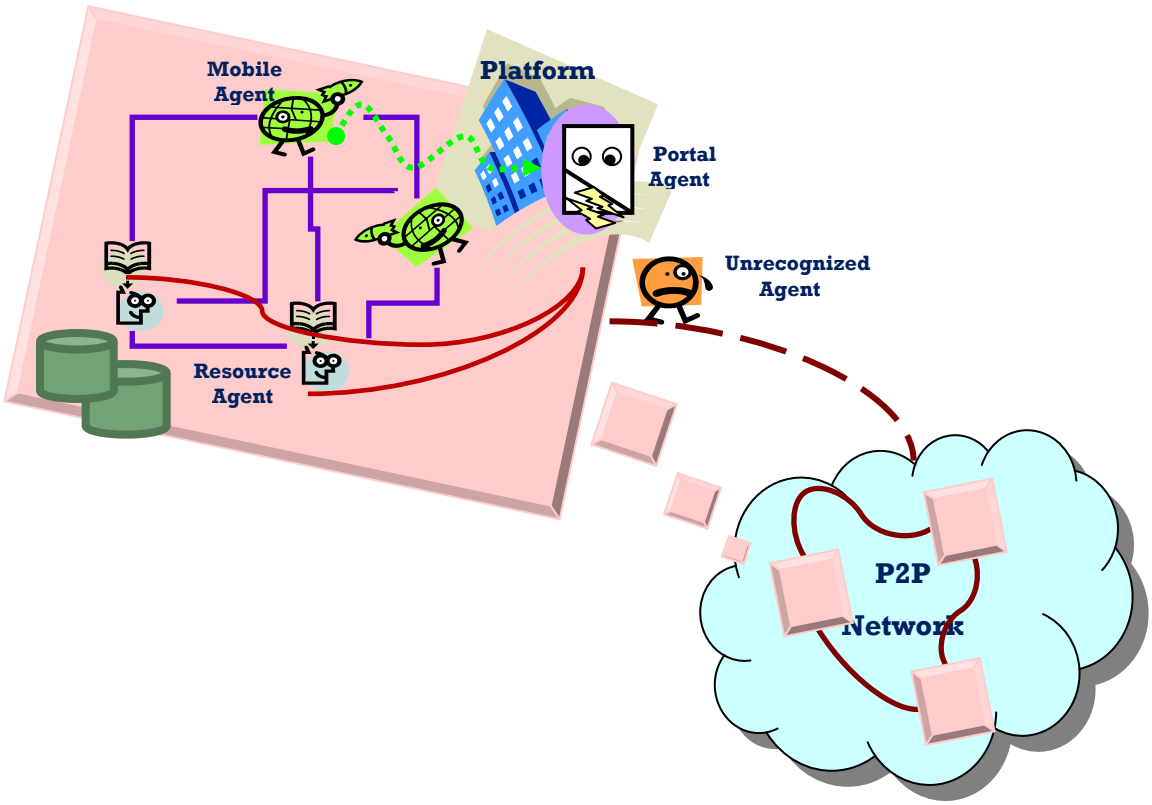


Figure 4-2: Agents' system architecture on a P2P network

The AP running on each site administers the transactions processing the data managed by the RDBMS of the corresponding site. The agent system consists of two types of agents: ***stationary*** and ***mobile*** that are issued by the application's transaction and managed by the AP.

Stationary Agents: these are of three types: ***transactional agent***, ***resource agent*** and ***portal agent*** that reside within a single AP of their creator throughout their life time.

- **Transactional Agents (TA)**: are agents that handle user transaction for data uploading and downloading. They are also responsible to synchronize the multi-level operations and issues compensating transaction if needed due to communication failure or inconsistent data modification.

- **Resource Agents (RA)**: are agents that register the data stored on the MMDB administered by the AP on the local DB server. The agents are managing indices of the available objects in a B+ tree data structure. The structure allows efficient retrieval of the media object on its search key.

- **Portal Agents (PA)**: are agents that are used as the gateway of the AP for communicating with the other APs. The PA are of two types that serve as incoming or outgoing portal:

  i. ***Arrival Port Agent (APA)***: upon arrival of an object, which could be MA or message, from other distributed nodes, the PA serving as ***arrival port*** accepts the object and lines it up in its queue until it is served by the AP incoming object handler. The AP which always listens for the portal queue pulls the object out of the queue on an incoming object event and passes it to its handler to give the required service by the object.

  ii. ***Departure Port Agent (DPA)***: on the other hand, when an object (MA or message) is required to be sent out of the AP, the PA serving as ***departure port*** lines up the object in its queue waiting until the port of communication is free. The AP that listens for an outgoing request event through its DPA serves objects in the queue by pulling them out of the queue and sending them to their

location. The objects select their destination site on their free will irrespective of the portal, hence their autonomous mobility.

The PA is also responsible for monitoring the security of the ASA through authentication of the migrated objects. When objects are sent out from a given AP, the departure portal agent (DPA) applies a private key encryption algorithm and seals them before they leave the AP. On the other end, the arrival portal agent (APA) upon receiving incoming objects applies the reverse process of the DPA and decrypts the sealed object to extract the actual incoming object. The objects are then authenticated by the AP for their type and requested action. Unrecognized object (agent or message) from the authentication is deprived of execution and deactivated from the platform.

Mobile Agents: the other type of agents and the most central point of the ASA is the MA. MAs are issued by the multimedia application transactions to represent the task within the transaction on the distributed sites. The MAs then identify their environment and perform the task set at the desired location. These agents are having a freedom of migration to support the mobility feature discussed in section 3.1 of this paper.

4.3 Transaction of the Distributed Multimedia Database

The two most common operations of the multimedia application in consideration are ***saving*** a multimedia object and ***searching*** for the multimedia object. In this section we will discuss the two operations as implemented in the proposed ASA.

4.3.1 Uploading Media Object to Multimedia Database

Multimedia file to be uploaded is first located by the application, decomposed and sent for insertion to the DBMS that are distributed across the nodes running with the AP middleware. During saving, the application issues transactions that will be processed in ACID property. Before the save operation of a large media file (a video file in AVI or MPEG format under the case study of this work), it is segmented to video parts in order to minimize the response time required for streaming the media upon retrieval.

For slicing or segmenting large media files AVI format, a third-party's video splitter[1] software is used in the prototype for testing.

Saving a media file can be issued by any of the sites running the AP, which in turn instantiate TA to handle the required tasks. The instantiated TA then proximate the user in generating the transaction of uploading the media object by creating and deploying required MAs. The saving transaction consists of the following steps in order.

1. The media file is read by the application, and the parameters of the media object are extracted. If the media file is a video file; for instance, extracted parameters include title of the video, format of the video, number of parts of video file, and other information that can be used later for the retrieval operation.

2. Then, MAs are created, assigned the video file parameters and threaded to run on every platform of the distributed system. The agents are concurrently deployed (migrated) to every site of the distributed environment and perform the insertion operation on their designated location leading to ***replication*** of the media object on every node. The MAs execute two levels of concurrent operations on every node:

    a. ***Object Level (Data Level)***: the media object (e.g. video) parameters are saved on the local database,

    b. ***Page Level (Index Level)***: the media object is inserted in the index tree of the corresponding RA in the AP, that later on will be used to access the object (retrieve efficiently) during search operation.

3. Reports are generated by the agents and sent back to the application initiating saving of the media object at the end of their execution.

4. When an AP receives confirmation message for replication of the video parameters, it initiates video part saving in a similar fashion. That is; step 1 through step 3 above are repeated for the part to be uploaded; replicating details of the part on every node in the distributed system.

[1] "AVI Splitter" from BRIZ Software is used for testing on AVI files for splitting large AVI files into an approximately equal size of 3MB files for uploading.

5. Upon receiving replication reports from the distributed site, the AP application either continues on the transaction or sends back agents for compensating transaction if failure or inconsistency of data is sensed.

    a. In case of inconsistency because of replication, compensating transaction is issued and sent to the defective sites to maintain the distributed sites in consistent state.

    b. If the reports are indicating success of the current part of the media object replication, the part is decomposed into frames and transactional MAs are issued to randomly ***fragment*** the video object frames across the sites. Saving of the video frames again consists of the data level and index level operations similar to the video parameters saving with only difference in type of data and operation.

6. Finally, successful ***fragmentation*** of the video frames will be reported to the initiating application.

A summary of the uploading transactions is depicted in the flow chart shown in Figure 4-3.

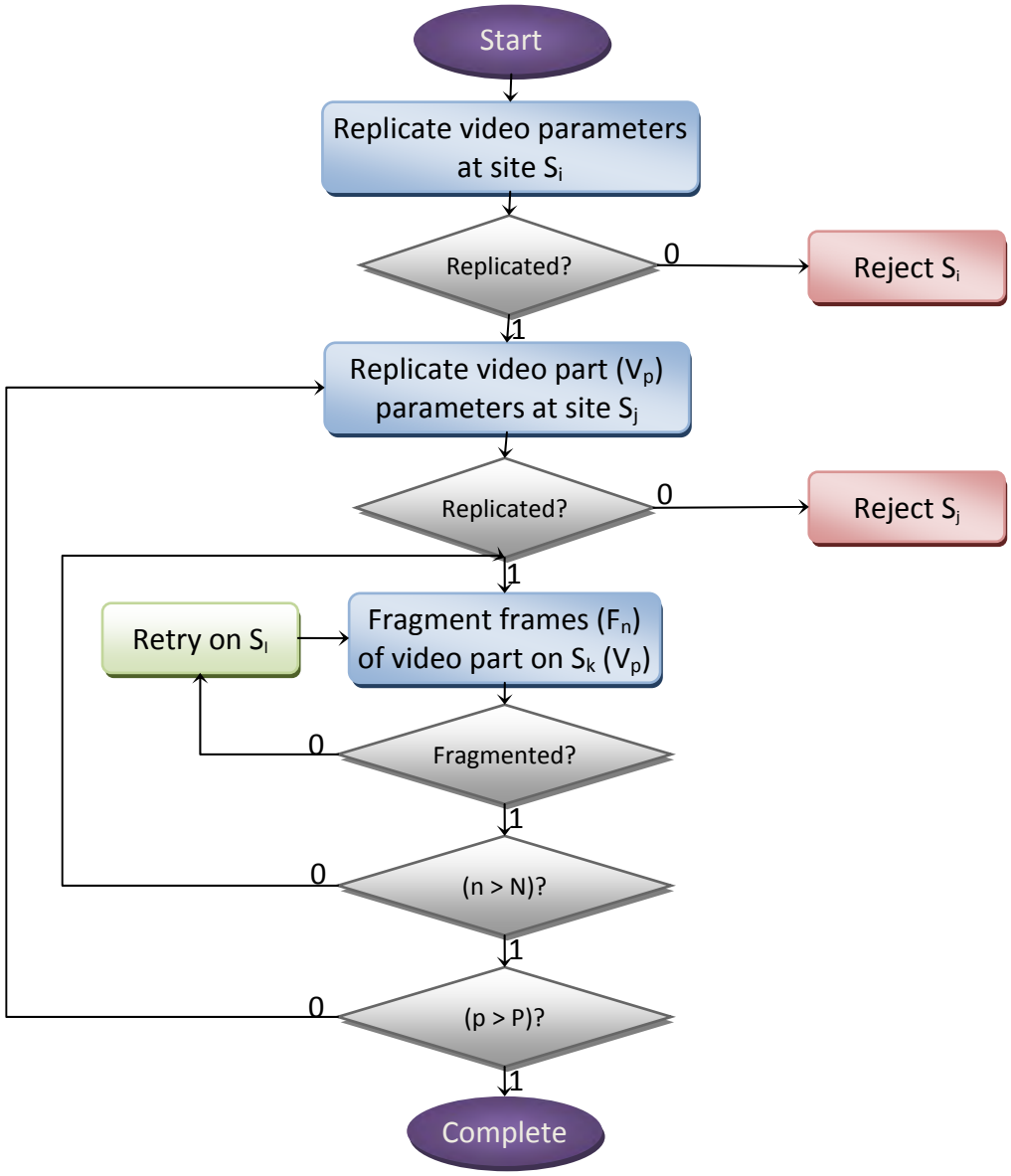


Figure 4-3: Flow chart for uploading transaction

Where; N is number of frames per segmented video part V_p and P is the number of segmented video parts.

Up on replications of data (video and/or video part parameters), a site S_i and/or S_j assumed to be working in the group of the distributed system may not respond for the replication. After sending awakening signal for replication within the wait period set by the platform, if the site is still not responding it is regarded as a failure site and rejected for fragmentation. This allows minimizing the losses that could occur during frame fragmentation. Whereas, in case of fragmentation the failed frames for confirmation are resent for any random site that didn't fail during data replication, as loss in data fragmentation is unaffordable as that of replication.

4.3.2 Downloading Multimedia Object

Once the data are uploaded to the distributed system, users require searching the data warehouse and accessing the data (downloading and streaming if supported). The proposed AP helps to generate a multimedia information search transaction in the DMMDBS with possible parameters. The AP application accepts searching parameter inputs (e.g. title of a video file) from the user and it initiates TA to handle the request of the user. It issues various transactions to search the data and stream it to the user.

The steps executed by the instantiated TA in searching of a media object (e.g. video file) from the DMMDB through the AP middleware are:

1. Search parameters are issued by an application running on the ASA platform. The search parameters are passed to the TA, which in turn organizes agents to do the job.

2. MAs are created and deployed by the TA on every active AP in the distributed system with the searching transaction.

3. The agents upon their arrival to the visited site, in search of the video file, contact the corresponding RA looking for the required file in the index; hence, executing index level transaction.

4. If an agent locates the media object, it sends out a report to the initiator application which in turn disseminates the information to the other agents in search.

5. Once the agents have located the video, they issue other TAs in retrieval of the video parts on their corresponding platforms. These TAs proximate the first TA in the requesting site and then extract the frames of the video part and deploy MAs to buffer the frames in the media requesting AP.

6. Then the AP searching for the media object, accepts these populated agents which process two level transaction as:

    a. ***Data Level***: in search of the frames and migrating located data to the user requesting the file, and

b. ***Presentation Level***: streamed video frames from multiple of MAs are used to generate the video file for presentation that is buffered on temporary file for read.

7. Once the first video part is located and completely buffered on the temporary file it is played to the user. While the current video part is playing, background transactions continue in buffering the next and subsequent parts in the same manner.

   By the time the foreground transaction of playing the current part is finished, the next part can continue as it would complete its buffering from the behind transaction. Occasionally, if there is significant network congestion, a media file may stop playing momentarily so that the buffer refilling completes for the next part.

The transactions for downloading a media file and streaming are summarized in Figure 4-4 flow chart.

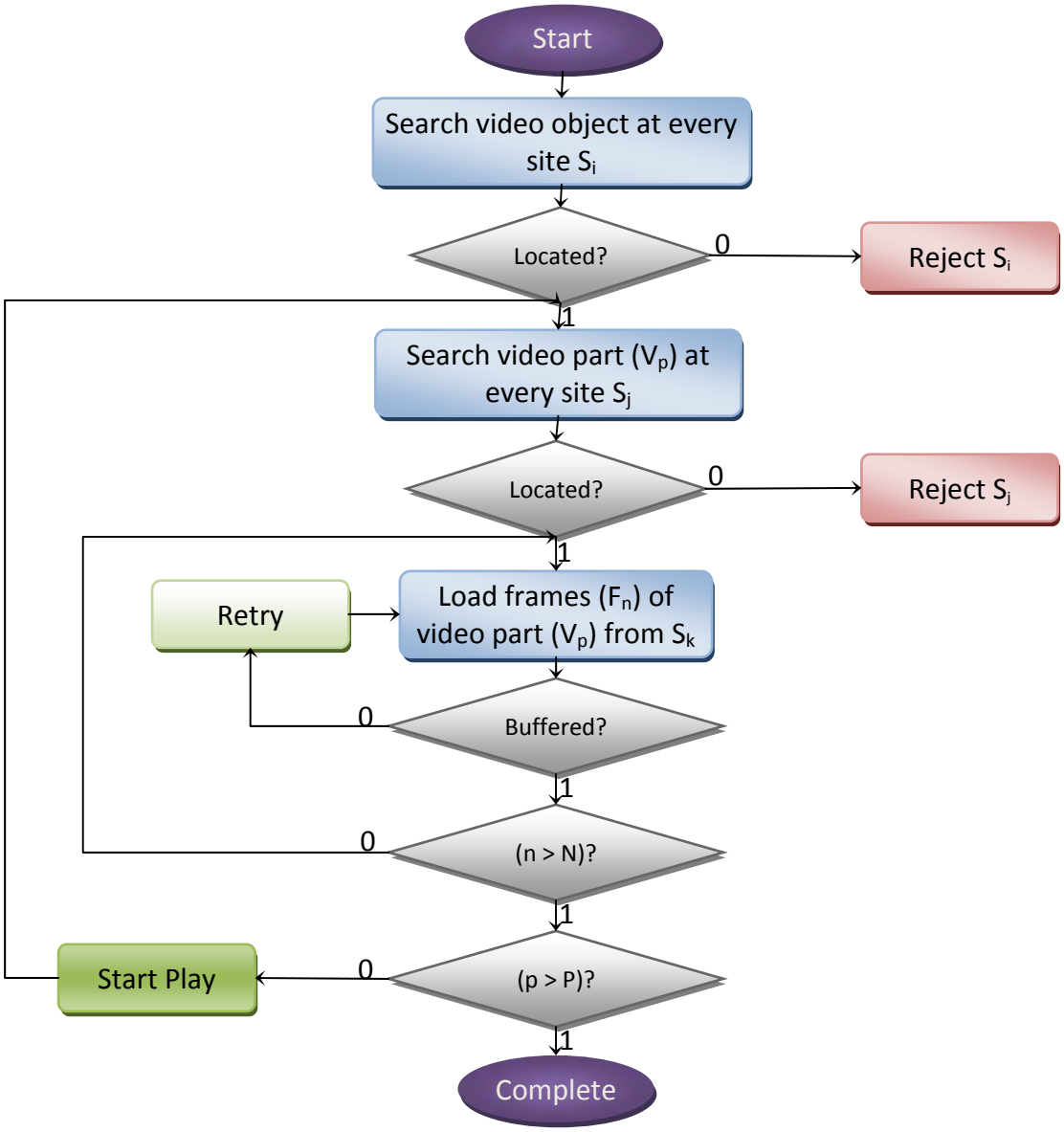


Figure 4-4: Flow chart for downloading transaction

Where; N is number of frames per segmented video part V_p and P is the number of segmented video parts.

In case the media file (the video object) is not located at site S_i, the active platform is rejected out from further request of video parts as the information is replicated in every active site during uploading. Likewise, if the replicated video part information is not located at site S_j, the site is removed from the active platforms lists for frame extraction. Upon loading request of video part's frames, if the number of located frames is matching to the total number of frames in the part, the frames are streamed and buffered for play; otherwise, the media file is treated as corrupted.

4.3.3 Indexing and Logging

Since the middleware is performing the transaction on behalf of the distributed MMDBMS, it needs to incorporate indexing and logging modules. While indexing improves the search efficiency, logging feature is used to maintain the integrity of the system.

Indexing

Index files are used for registering resources available at the platform. These index files are managed by the RAs of the corresponding media objects. For instance, in the case study the video and video part parameters are indexed into two RAs of the AP.

The B+ tree data structure is used for indexing allows the records to be located efficiently without the need for database access from the storage media. The structure sorts data in a way that allows efficient insertion, retrieval and removal of records, each of which is identified by a key. It is a dynamic, multi-level index, with maximum and minimum bounds on the number of keys in each index segment. All the records in the B+ tree are stored at the lowest level of the tree; only the keys being stored in the interior blocks.

Upon a media uploading the index file is updated through an agent executing insertion to the RA. On the other hand, during downloading the index lookup agent contacts the RA to locate the media object through its search key.

Logging Transaction

During media file uploading as seen under section 4.3.1 multiple transaction are issued at the three different layers. These transactions are needed to be logged. A log management is required to manage the logging transaction. The log files can be used at the time of recovery after a failure occurs on the site issuing the transaction.

For this reason, every save and read job is logged on the platform initiating the job. As depicted in the Figure 4-5, two level log files: index-level log and data-level log are used.

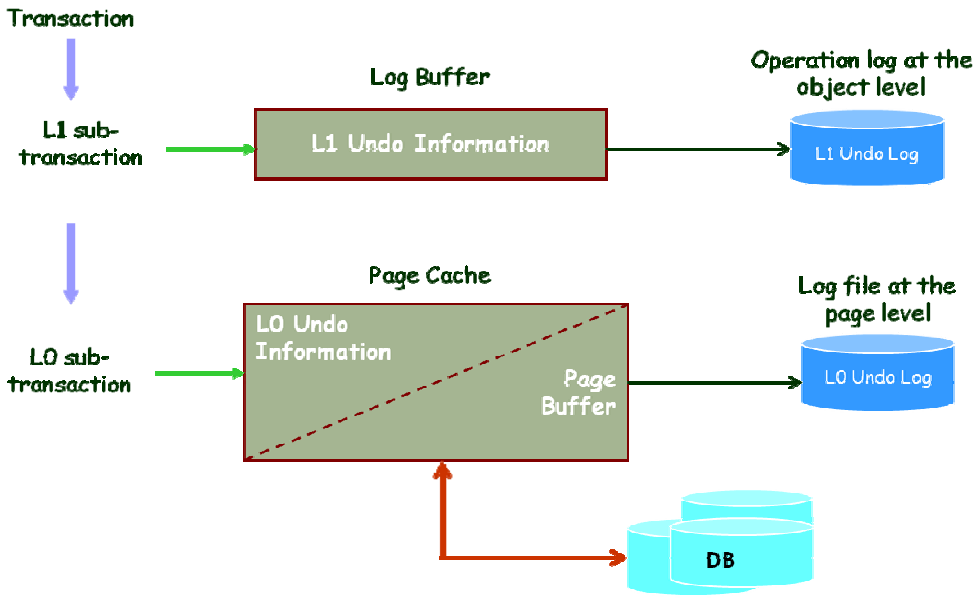


Figure 4-5: Transaction logging activity

Index-level logging: the level, L1, logging indicated in the figure is the index level logging that is built on fly of indexing transaction that is not actually stored on the database management system. This logging is required to track the integrity and consistency of the middleware with respect to the database management system.

Data-level logging: the level, L0, logging is the data level logging that logs every transaction associated with data read/write executions issued to the distributed nodes. The logging keeps the integrity and consistency of the DDBMS in data replication and fragmentation.

Both the index-level and data-level logs are consulted for every compensating and undo transaction during failure or inconsistency.

4.4 Agents Activity

The transactions stated in the previous section are organized and executed by employing the stationary and MAs. These agents are managed by the AP of the application running on the distributed sites. The activities of the agents for handling the two basic transactions are summarized under the agents activity diagram Figure 4-6.

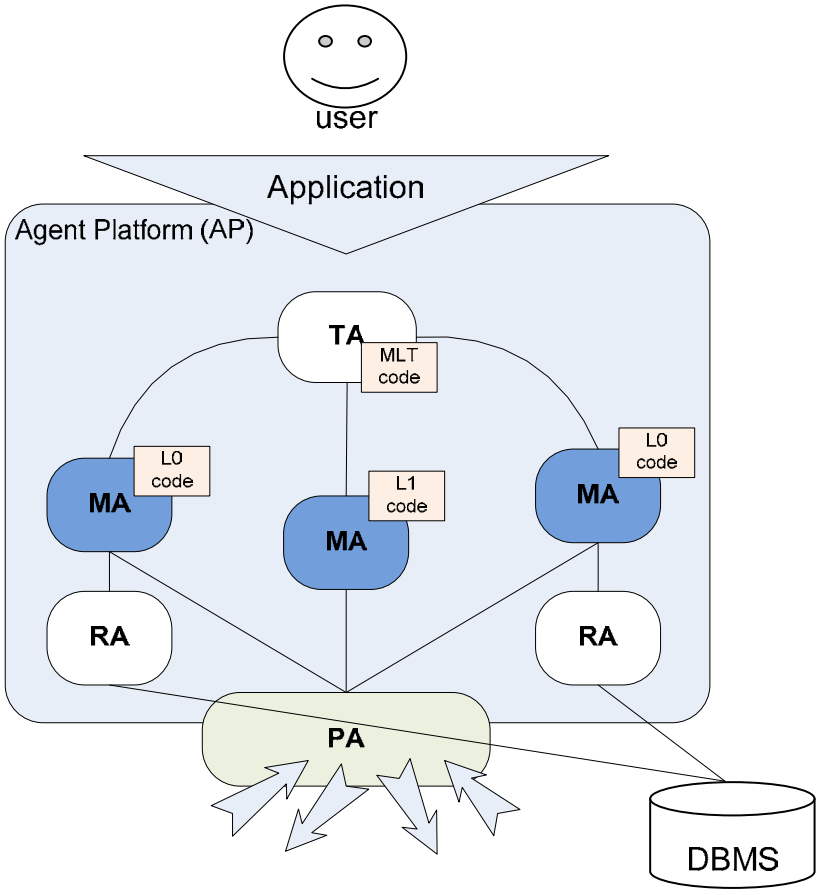


Figure 4-6: Agents' activity diagram

The user is interacting to the DMMDMS through the application running the AP. As indicated in the figure, the TA is the center of all the transactions for handling the MLT codes for both uploading and downloading operations. It generates MAs to handle different level transactions at the distributed sites. The MAs are then on their autonomy to contact the RAs and/or the PAs. While RAs are the one that manage indexing of the resources avail on the AP, the PAs implement the security features of the middleware during MAs migration.

5 Materials and Methods

As mentioned earlier in the text, a prototype is built to investigate the performance of the proposed architecture. The MAs in the system are required to migrate from site to site and execute MMDB transactions at their destination. For their mobility network programming and process communications are required.

In this chapter, we will describe the major tools and methods used in the development of the proposed architecture supporting the MMDB transactions, network programming and media presentation.

5.1 Database Implementation

The test MMDB is designed and implemented in a standard relational database management system (RDBMS). SQL Server 2005 database management software is used for the implementation.

In the MMDB; as it is implemented through the RDBMS, the multimedia data are stored as BLOBs. For searching and annotating parameters of the media object (such as, title of a video file) are stored as text and standard data types supported by the relation database. To provide the object nature of the multimedia files a media module for the object representation of the stored relational data is implemented at the middleware. The module is programmed in Java with OO approach. This makes the MMDB design to be object-relational design. Stored procedural transactions are used in the RDBMS to be executed by the objects in the middleware for data insertion, read and modification.

5.2 Java Implementation

After the database is designed and implemented in RDBMS, a prototype middleware is implemented in an OO design. Since Java is a suitable language for an OO design in distributed application, the middleware is developed in Java, jdk 1.6. For the Java project development NetBeans IDE is used which allows for the design of the application user interface.

5.2.1 Objet Serialization

In Java programming, one can write an object of any kind to a stream. This could be to a disk file or to a Server via a Socket. Also, it can read an object of any kind from a stream. For sending out the MAs and messages across the distributed sites streaming of the objects is done through sockets opened on the communicating sites.

The object serialization provides the ability to read or write an object as raw byte stream. It allows Java objects and primitives to be encoded into byte stream suitable for streaming to a transmission media. To do so, the ***ObjectOutputStream*** and ***ObjectInputStream*** classes are used. The constructor for the first takes an *OutputStream* argument; the constructor for the second takes an *InputStream* argument. Files can also be opened to get these, but for the case under study; client-server socket communications, the ***Socket*** class methods are used to obtain these I/O streams.

OutputStream

The output stream is used to send out objects through prepared socket. The stream is opened through the socket as follows:

```
ObjectOutputStream objOut = new ObjectOutputStream(socket.getOutputStream());
```

The *objOut* object wraps the stream object specified as a constructor argument *socket.getOutputStream()*.

Then, the *writeObject()* method of the ***ObjectOutputStream*** is invoked to write the object to the socket. Hence, the stream marshals the written object into binary streams and sends it out the site waiting for the object through the opened socket.

```
objOut.writeObject(outgoingObj);
objOut.flush();
```

InputStream

Likewise, the object input stream is instantiated from service socket opened from an AP acting as the server for the request coming from other sites.

```
ObjectInputStream objIn = new ObjectInputStream(serviceSocket.getInputStream());
```

The incoming object is read through the *readObject()* method of the *objIn* object of the ***ObjectInputStream*** class. Once the object is read its reference is casted to the actual type of the object which un-marshals the binary stream to the object.

```
objIn.readObject();
MobileObject obj = (MobileObject) objIn;
```

Output streaming and input streaming of the ASA is done through the two types of portal agents DPA and APA.

5.2.2 Java Multimedia Framework

The Java media framework API (JMF) is a large and versatile API that enables audio, video and other time-based media to be added to applications and applets built on Java technology. This optional package, which can capture, playback, stream, and trans-code multiple media formats, extends the Java 2 platform, standard edition (J2SE) for multimedia developers by providing a powerful toolkit to develop scalable, cross-platform technology.

A complete reference implementation, JMF 2.1 is enabled to do almost everything imaginable with multimedia. The JMF is used to play various multimedia files in a Java applet or application. The formats supported include AVI, MPEG and QuickTime (MOV). The framework can also be used for streaming media from media streaming server.

5.3 Implementation Modules for the Middleware

Since the middleware is implemented in Java, it ideally runs on any platform supporting the Java virtual machine (JVM). The middleware is basically organized into five modules *Agent and Agent Platform module*, *Task module*, *Resource module*, *Security module* and *Media module*.

5.3.1 Agent and Agent Platform Module

The **Agent and Agent Platform module** is the center of the ASA. It comprises the class definition for both the stationary type and mobile type agents. It also defines the platform and its functionalities.

Some of the major classes defined in this module include: *Agent*, *MovableObject*, *MobileAgent*, *PortalAgent*, *ResourceAgent*, *TransactionalAgent*, *AgentPlatform*, *PFResgistraion*, and *Message*.

```
/**
 * Agent class to be extended by every type of agents.
* @author Betiglu
 */
public abstract class Agent extends Thread
                implements Cloneable, Serializable {

    /**
    * Clone the agent objet
    * @return cloned object
    */
    public Agent clone() { …          }

    public boolean talkTo(Message msg, Agent partner, int port) { …                    }

    public void dispose() throws Throwable { …                }

    // …

}

/**
 * Mobile Agent class used for carrying out the mobility nature of the system
 * @author Betiglu
 */
public class MobileAgent extends Agent implements MovableObject {

    /**
    * Method for migration: Connect to the agent platform to migrate through its portal
    * @return boolean status of registration
    * @throws java.io.IOException
    */
    public boolean migrate() throws IOException { …                }

    public boolean retract () throws IOException { …                }

    /**
    * Method used to run the process of the agent
    */
    public void run() { …             }

    // …

}

/**
```

```
 * The Agnet PlatForm
 * @author Betiglu
 */
public class PlatForm {

    /**
    * Run the platform to listen for various communications
    */
    public void run() { …              }

    // …

}
```

The AP upon instantiation, initiates the two types of stationary agents:

i. ***Resource Agent***: the RAs are initiated to register the resources available on the platform. The agents read the corresponding RDBMS and build a resource index tree within the ASA middleware. These agents are consulted during media retrieval.

```
// Instantiate the resource agents for indexing
// Video objects
ResourceAgent vidResourceAgent = new ResourceAgent(new MediaObjectIndex())
vidResourceAgent.getTask().setCode(new LoadVideoIndex());
vidResourceAgent.setOwnerPF(this);
vidResourceAgent.start()
```

ii. ***Portal Agent***: two of these agents are initiated to present arrival and departure ports to the AP. For the arrival port, an APA is initiated waiting for the platform to activate it upon incoming objects. DPA is issued to open a socket and be ready for agents and messages to be uni-casted through.

```
// Instantiate the portals for departure and arrival
// Arrival portal agent for the platform
PortalAgent arrivalPortal = new PortalAgent(PortalType.ARRIVAL);
arrivalPortal.setOwnerPF(this);
arrivalPortal.start();
// Departure portal agent for the platform
PortalAgent departingPortal = new PortalAgent(PortalType.DEPARTURE);
departingPortal.setOwnerPF(this);
departingPortal.start();
```

The module also defines the messaging scheme used by the agents and the platforms for communication and registration. When an AP is run, it registers itself to the other active APs in the distributed system through the platform registration unit in this module. Up on deactivation of a platform the registration is canceled by the same unit.

5.3.2 Task Module

The agents are executing transactions that are defined and constructed under the **Task module**. This module defines the ordinary and compensating transactional codes which can also be MLT codes such as video file decompositions and compositions and video frame saving and searching.

The tasks in this module are designed to be serialized and the class files to be sent as the code part of an MA. With the Java nature of interoperability the class files are executable at the destination site through the MAs carrying them to the destination. Thus, the tasks can be originated at any site and executed on any site requested by the transaction.

5.3.3 Resource Module

The resource module is containing of two sub–modules for indexing and logging:

i. ***Index sub-module***: the index module is built to present the indexing structure required by the RAs, and

ii. ***Log sub-module***: is a module used for logging of index level and data level transactions. The module logs every transaction on a temporary storage used for undo and redo operation at the time of recovery if needed. In normal operations (free of failure) the log module is redundant.

5.3.4 Security Module

The **Security module** consists of authentication and authorization operations done through encryption and decryption functionalities. The Java cryptographic security classes are implemented to encrypt and decrypt objects in the data encryption standard (DES) cryptography algorithm. The security feature is applied by the PAs: encryption by the DPA and decryption by the APA.

5.3.5 Media Module

The **Media module** is a module where the functionalities of the media object is supported. It defines decomposition and composition algorithms. And also it holds the implementation of media play functionality. It uses the JMF 2.1 for the presentation.

It is in this module that the multimedia data is also mimicked from the relational data to its object nature.

6 Analysis and Testing

The proposed architecture discussed in the previous chapter is implemented in a prototype application. The prototype is tested for AVI and MPEG video files uploading and downloading. The result of tests performed on the designed architecture and its comparison with a legacy RPC test program for the same task is presented and discussed in this chapter.

6.1 Architectural Analysis

The proposed architecture is tested on a distributed environment of multimedia data managed by standard database managements systems.

The RDBMSs are installed and they are managing their data on their own administration modules, the architecture of the distributed database being a federated (heterogeneous) multi – DBMS architecture. The prototype is tested merely with multiple SQL Server 2005 database management system. But, the application can easily be adapted for any standard RDBMS with a slight modification on the connection sub-module of the platform.

Every AP in the distributed system is of equal importance and supported with same functionalities. Each site is set with an AP as the middleware and RDBMS for data management. This makes the distributed architecture to be a P2P network. The agents' job in the platform is then coordinated through the different class of agents which may be assigned for different tasks.

The TAs assume the platform to be centralized as it coordinates the MAs. However, when the MAs are deployed in the system they are doing their task in a distributed fashion. Moreover, the task of TA is supported with the portal and resource agents in a distributed P2P manner as well. Hence, within the platform distributed coordination of agents is achieved, the architecture for the MLT being distributed platform – distributed agent.

The decentralized platform – decentralized agent MLT approach discussed in section 3.4 is achieved through three levels: index level, data level and presentation level. Besides that, the basic MA requirements of *autonomy*, *social ability*, *pro-activeness* and *mobility* are implemented in the architectural design. The

MAs are secured through the security module that enforces authentication and authorization of agents.

6.2 Testing

The prototype is tested against two classes of test: System Performance test and Comparative test with the legacy RPC application for the same task. The determining factor for the ASA performance improvement is set to be the QoS for online downloading transaction. The uploading transaction is on the other hand possibly an offline transaction with less constraining temporal requirement.

6.2.1 Performance Test

The performance of the prototype is measured in terms of QoS functionalities of required bandwidth as byte transfer rate and response time. The performance is tested with and without video segmentation for downloading and streaming within the faculty Intranet infrastructure of 100Mbps bandwidth. Moreover, after the prototype is pruned for video segmentation, the uploading transfer rate is tested and optimized.

Test without Video Segmentation

First attempt was made to distribute a video file without segmentation. Since a temporal constraint is more crucial during media file download; after the complete uploading of a media file, the transfer rate for downloading video files in the range of 275 KB to 9MB is tested. The test is performed on single–node and two–nodes distributed system. The result as given in Appendix A is depicted in the video byte transfer rate versus video size curve of Figure 6-1.

The curve of the downloading transfer rate in the figure is approximated to a logarithmic increase as the files size increases. The increase in the transfer rate is achieved since the pay load of the over all system gets optimized as the file size increases. As it is shown in the figure approximately at a 3MB size of a video file, the curve flattens to a constant value, the agent system reaching to its optimal transfer rate. Then on ward, the increase in the transfer rate is less to the extent being constant value (for a single-node distributed system) considered for the optimal transfer rate.

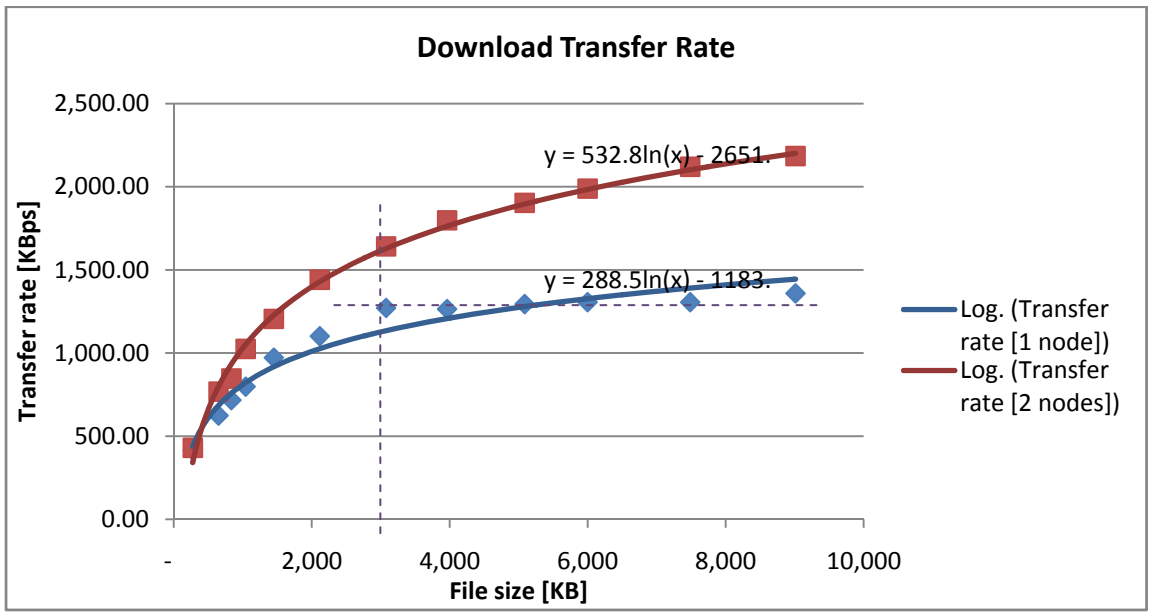


Figure 6-1: Downloading transfer rate , without segmentation

The logarithmic transfer rate produces an approximately linear response time shown in Figure 6-2. The linear characteristic makes the response time to be unbearable for a user requesting a large media file downloading. This calls for segmentation of a large video file into video parts that can be streamed within an acceptable response time.

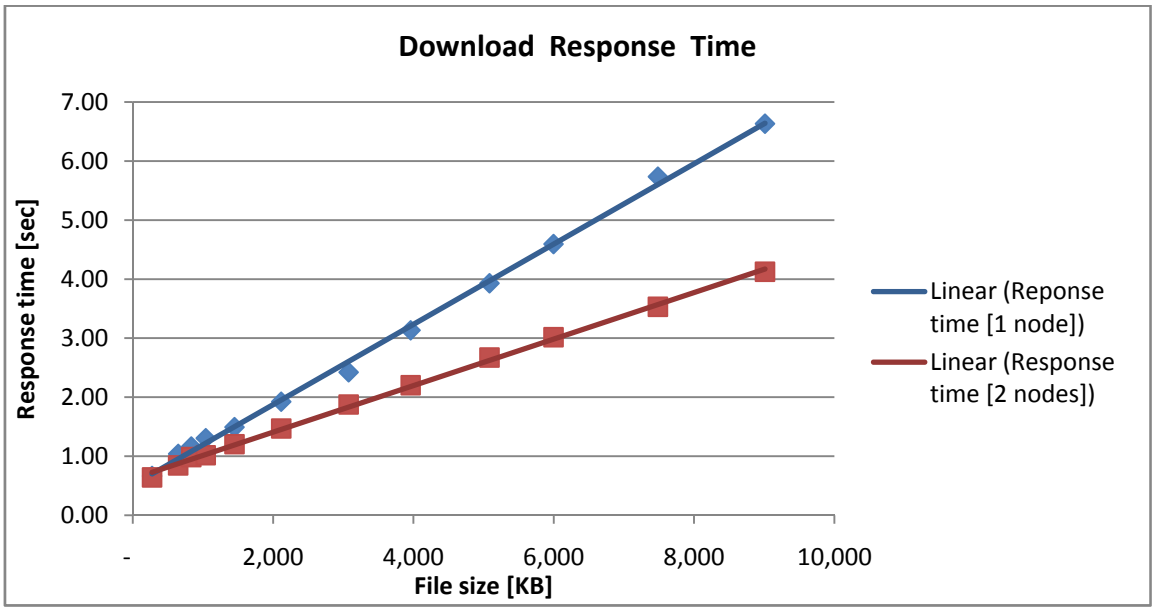


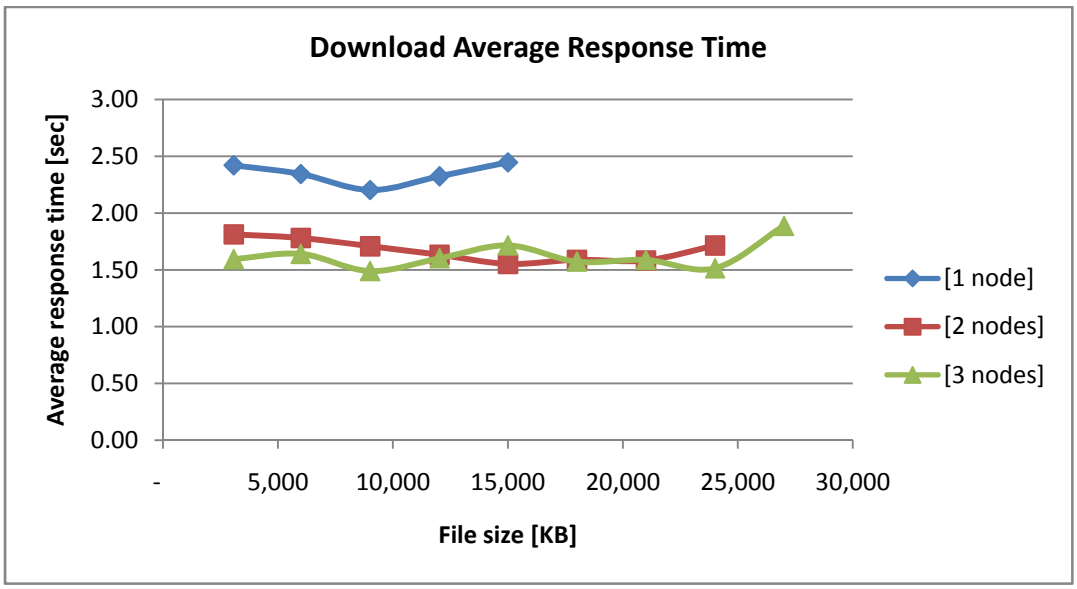
Figure 6-2: Downloading response time, without segmentation

Testing with Video Segmentation

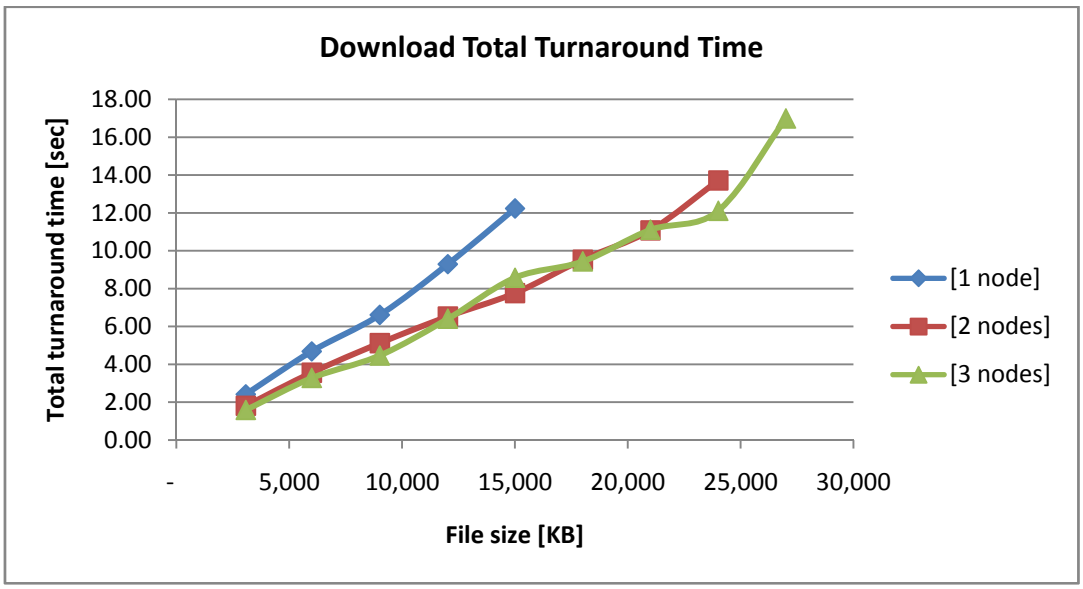
Testing video file downloading without segmentation has shown a logarithmic increase transfer rate. For a single-node distributed system it has exhibited averagely a constant transfer rate of 1,250KBps for a media file larger than 3MB. The rate is in average 2MBps for a two-nodes distributed system. Thus, large media files are segmented into an average of 3MB size during upload time.

Then, the bandwidth requirement is optimized by considering the average transfer rate for the video play-out rate. The segmented video files are found to be in a range of 12 to 25 second duration. Considering the average segmentation size the play-out rate is in average 270KBps. Since the AVI video files used in the prototype are found to be in average 270KBps play-out rate, a grace period is introduced in every range of frames buffering. The grace period can be selected in such a way that it prunes the media frames transfer rate to a considerable range of the average play-out rate. However, in a practical case the grace period could be avoided to combat the effect of network congestion.

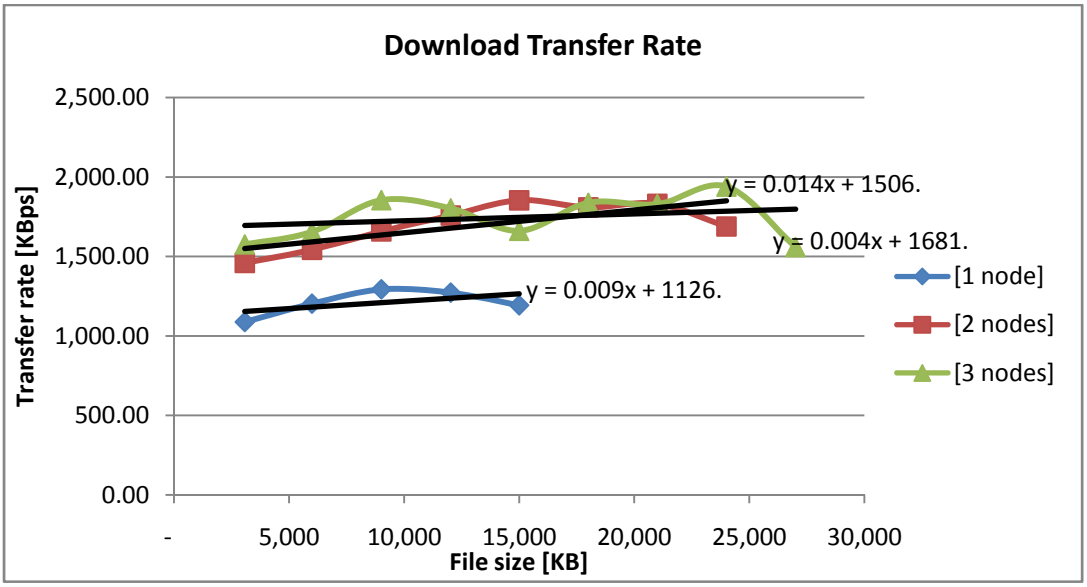
Figure 6-3 shows the downstream average response time, total turnaround time and transfer rate curves for video file downloading in a distributed system with 1-, 2-, and 3- identical site(s). The curves are obtained from the result shown in Appendix B.



(a) Avarage response time



(b) Total turnaround time



(c) Transfer rate

Figure 6-3: Downloading results with segmentation for 1- 2- and 3- nodes

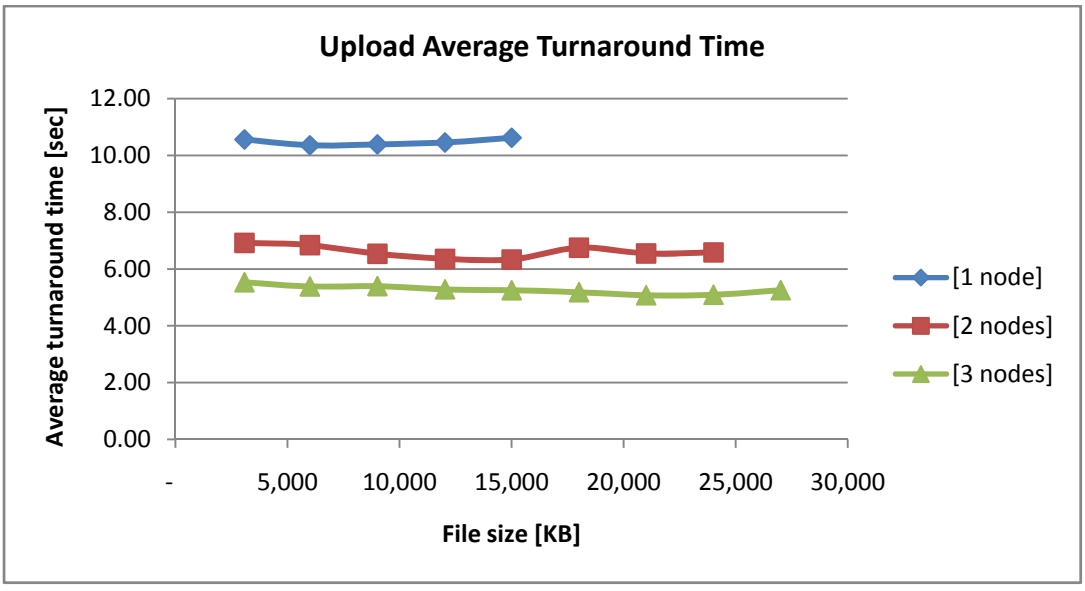
While the average **response time** shown in the Figure 6-3(a) is measured as the time elapsed before the video part is readily available for playing, **jitter** (the variance of the transfer delay) is used to measure the delay between successive parts streaming. It is measured as time elapsed between the time a video segment (part) downloading is completed and the video part playing issued after the previous playing part is completed. The jitter is minimized to create the impression of continuous play in a video presentation by completing part

buffering in due time. With the ideal test, it is possible to achieve a zero (considerably negligent) jitter.

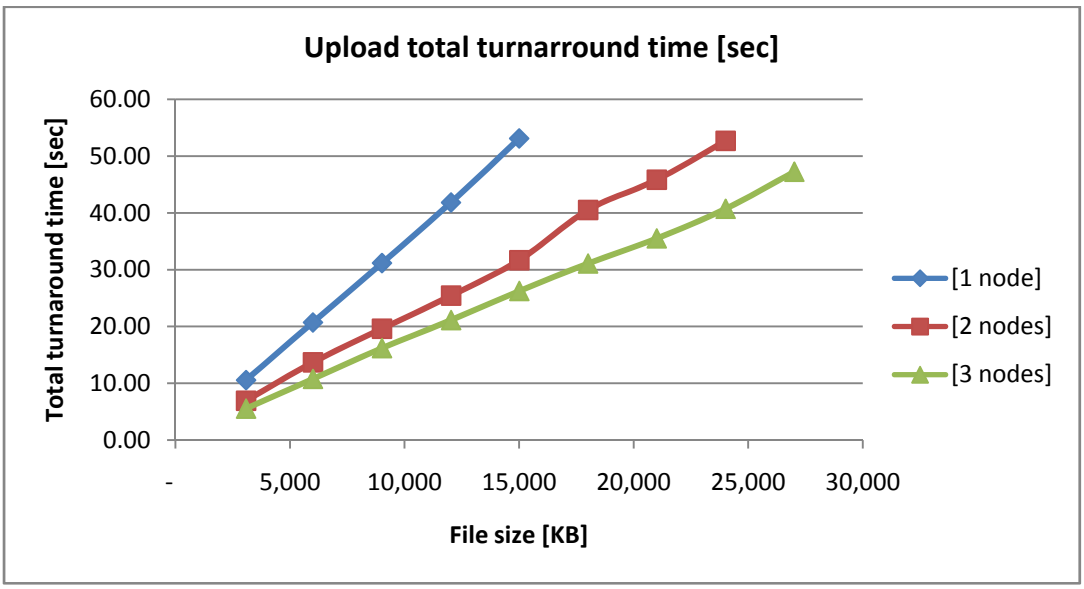
Figure 6-3, then depicts that in average the video files are completely streamed to the requesting node within 1.67 and 1.62 seconds response time with an average transfer rate of 1.70MBps and 1.75MBps for nodes of two and three distributed system, respectively. This makes the jitter to be zero for a video file segmented in a range of 12 to 25 seconds; hence, by the time the currently playing video part is complete the next video part is ready for play.

Though; the downstream transfer rate is saturating at an averagely 1.7MBps for multiple nodes, the upstream transfer rate improves over multiple nodes of the distributed system. Figure 6-4 shows the upstream turnaround time and transfer rate for video file uploading in a distributed system with 1-, 2-, and 3- identical site(s).

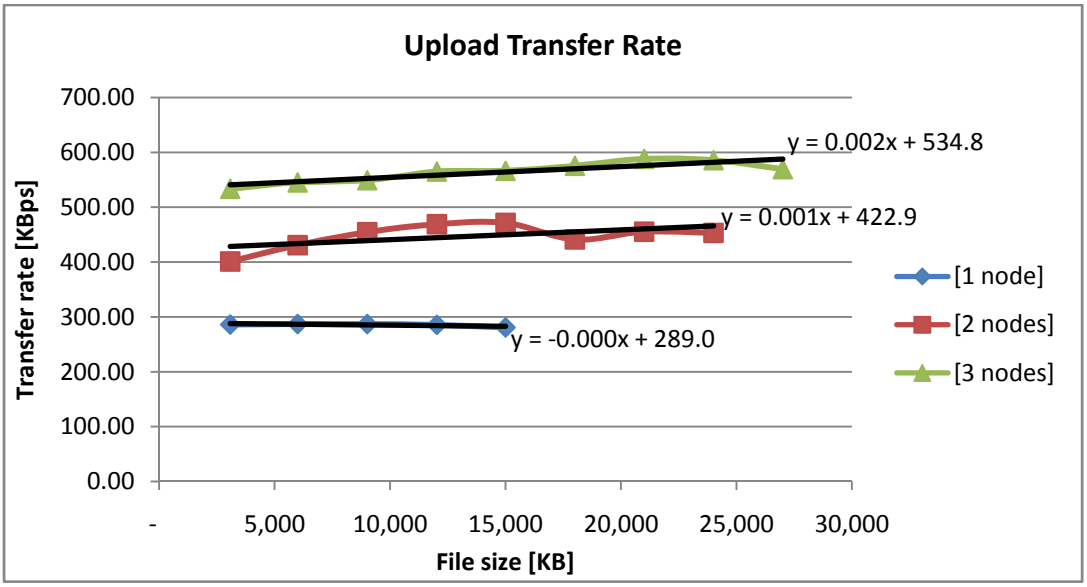
For the upload transactions the turnaround time is measured as the time taken for completion of the segmented video files uploading. Figure 6-4(a) shows that the average turnaround time for uploading the 3MB video parts is approximated to constant value of 10.5sec, 6.6sec and 5.3sec for 1-, 2- and 3- nodes distributed system.



(a) Avarage turnaround time



(b) Total turnaround time



(c) Transfer rate

Figure 6-4: Uploading result with segmentation for 1- 2- and 3- nodes

Corresponding to the constant average turnaround time for upload, the total time taken for uploading the media files is naturally linear. The upload (upstream) transfer rate, as shown in Figure 6-4(c) is improved as the number of nodes in the distributed system increases. In average the upstream transfer rate is found to be 285KBps, 447KBps and 564KBps for 1-, 2- and 3- nodes distributed system respectively.

Observation

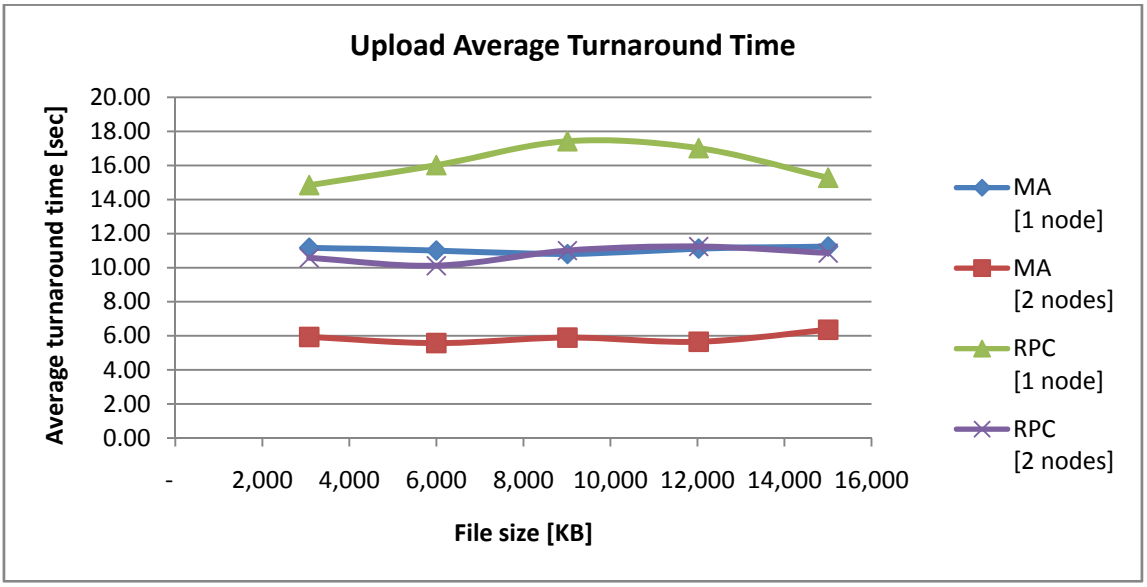
The prototype has exhibited increased media frame drop for uploading as the size of the files increase. This is from the fact the network will be over-flooded from the agents moving from the source to their designated nodes.

6.2.2 Comparative Test

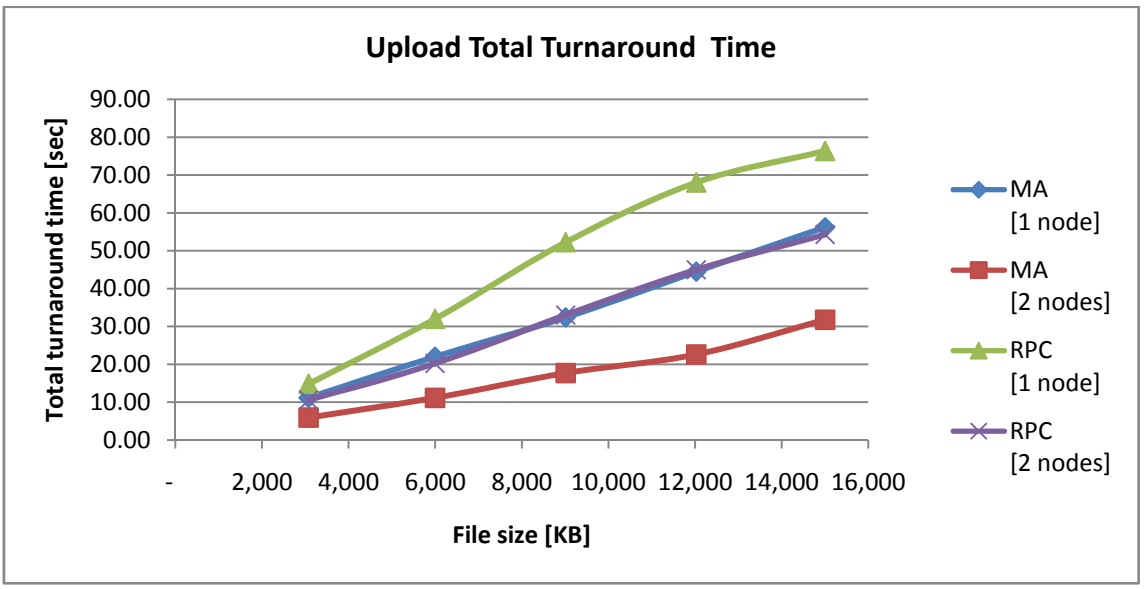
The legacy RPC test application is built for the same task achieved through the prototype of the proposed architecture. The prototype and the test RPC application are tested for segmented AVI video file uploading and downloading on single – node and two – nodes distributed system.

The two systems are compared in their transfer time and response time for both operations. For the comparative test unrealistic P2P network infrastructure is considered. The results are presented in Figure 6-5 and Figure 6-6.

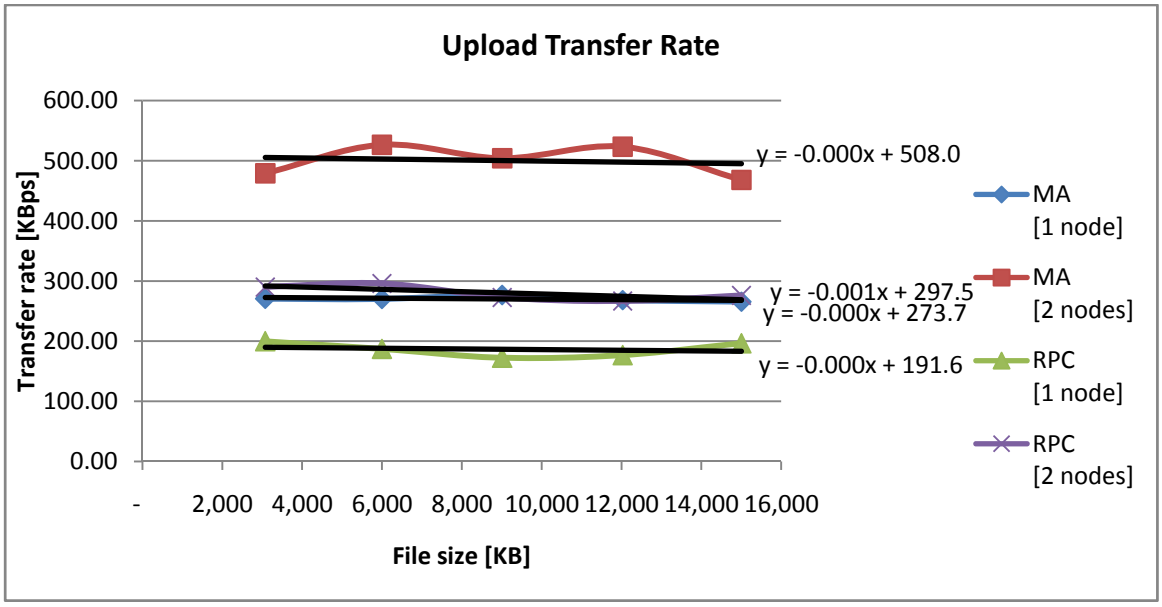
For the uploading test the legacy RPC application has an average transfer rate of 186.45 and 280.32 KBps for single-node and two-nodes distributed system, respectively. The transfer rate comparison curve in Figure 6-5(c) shows that the MA-based architecture gives 44.94% and 78.46% improvement on the upstream transfer rate. The figure also indicates that, the transfer rate is averagely constant for both test applications so as the average turnaround time for uploading video parts.



(a) Average turnaround time



(b) Total turnaround time

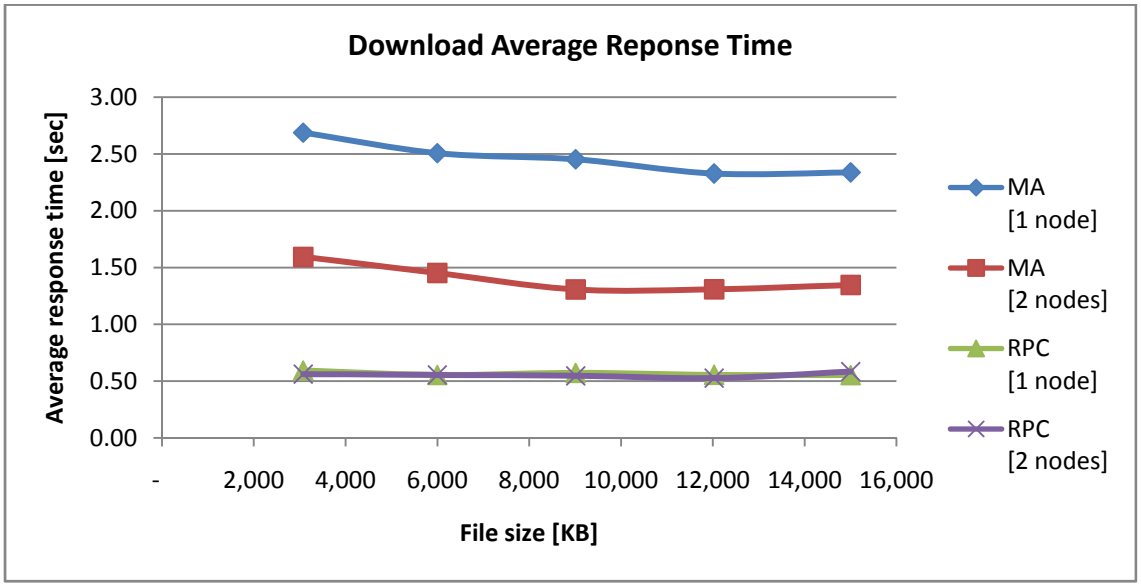


(c) Transfer rate

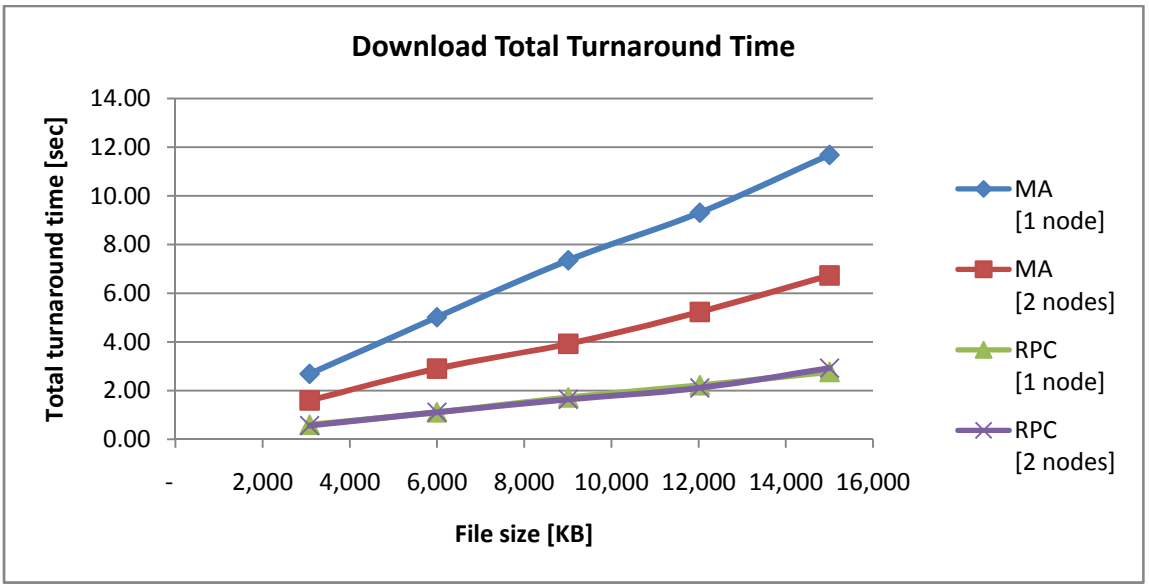
Figure 6-5: Comparative test on video uploading

Unlike; to the upstream transfer rate improvement of the MA-based architecture, the downstream transfer rate which is found to be 1.15MBps and 1.96MBps for the the single-node and two-nodes distributed sytems has shown 78.03% and 78.46% decline; respectively. Though the downstream is signifacntly lowered, Figure 6-6(c) depicits the increasing rate (slope of the approximating function) is slightly improved.

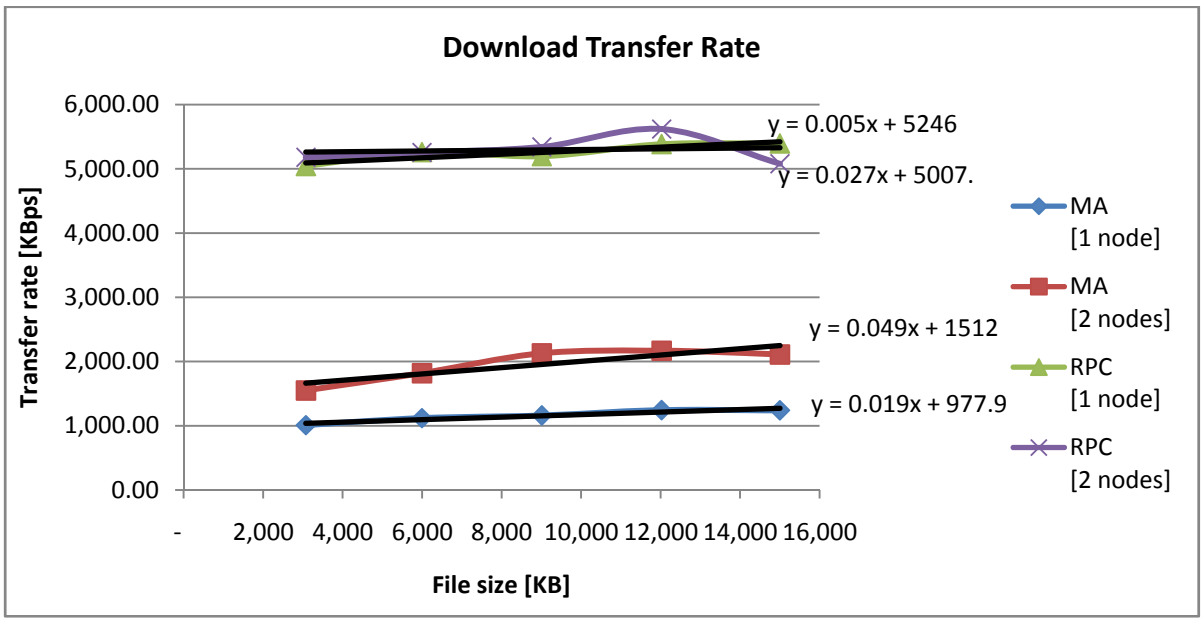
As shown in Figure 6-6(a), the average repsone time for the legacy application is found to be considerably small 0.56sec but constant for both single-node and two-nodes distribution. On the other hand, the MA-based architecture prototype has shown a 43.07% improvement from 2.46sec to 1.40sec for the two-nodes distributed system over the single-node distributed system.



(a) Average response time



(b) Total turnaround time



(c) Transfer rate

Figure 6-6: Comparative test on video downloading

Observation

The legacy RPC application has exhibited a connection time out to the database server for the remote execution of the stored procedure unlike to that of the MA-base architecture. The MA-based architecture is free of such drawbacks as the codes to be executed are sent to where the data is located for execution.

6.3 Summary

Multimedia data, such as images, audio and video are increasingly popular almost in every today's information dissemination as compared to that of textual data. However, various designing issues are raised for managing multimedia data in DDBMS which is often done through MLTs. Some of the issues studied in this paper includes: *Long transactions and multi-level/nested transactions, Composition and decomposition of multimedia objects, Persistence object, Indexing* and *Security*.

The MLT of a distributed MMDBMS is basically done on three layers: ***interface layer*** for presentation, ***object composition and decomposition layer*** and ***storage layer***. The latter two layers are done in index and data levels of the MLT. Often, the distributed system is a P2P network where every site is contributing for data

storage and querying. The researcher designs a middleware based on the MA architecture for handling the MLT.

The MA system is built with the very nature of mobility required by any distributed system. ***Mobile agents (MA)*** is composed of transactional code, set as a task to be executed by the agent on the data at the destination of its request. A platform is design to manage the MAs as well instantiate stationary agents of the system. There are of three types of stationary agents managed by the system. ***Transactional Agent (TA)*** is an agent that is responsible to manage the MLT through the MAs. The second type, ***resource agent (RA)*** is an agent that registers and manages resources of the platform for efficient data retrieval through indexing. The other type of stationary agent is the ***Portal Agent (PA)*** that is responsible for the mobility of MAs, and agents and platforms communications. The PAs are also responsible for the authentication and authorization processes of the security module.

A trustworthy agent system is built on an unrealistic infrastructure that assumes agents are only migrating to confined and managed platforms. The systems addresses security treats on the agents and the platform through a combined feature of *safe code interpretation, itinerary recording and path histories, encryption, authentication* and *authorization*. Besides that, the system achieves the concurrency requirement of a transaction through a threaded execution of mobile and stationary agents.

The results in section 6.2 show that the MA-based architecture has an average of 1.7MBps downstream and 447KBps and 564KBps upstream transfer rates for a distributed system of two and three nodes; respectively. These results are acceptable to the temporal constraints of the media files in consideration having a play-out rate of 270KBps.

7 Conclusions and Recommendations

7.1 Conclusions

The agent's system architecture (ASA) is built with a center of its application being the MLT of distributed MMDB handled by the TAs. The performance test of the ASA shows that it completes the downloading and streaming transaction within the temporal constraint set by the environment.

As the researchers in (Eid, et al. 2005) have pointed out, the MAs architecture tested in here is successful in distributing a multimedia data through the MLT management brought into the middleware. Though it is difficult to make a figurative comparison with the previously done researches in the area, this work has contributed by presenting MA based architecture for the same.

In general, the ASA manages the MLTs that enable:

- **Distributing media object**: the architecture is proven to distribute the media object through replication and fragmentation successfully maintaining the consistency and integrity of the distributed database,
- **Decomposition/composition**: the major MLT tested being decomposition and composition operations of the uploading and downloading transaction, the prototype proven the applicability of the architecture in the same, and
- **Persistence presentation**: the persistency requirement of the multimedia data is achieved through buffering and streaming of the segmented media file. Size based video partitioning and frame decomposition/composition is used. Persistence of media presentation is achieved through the video partitions played consequently.

The novel aspect of MLT management is presented in the ASA. The ASA encompasses data, index and presentation transactions. In addition, it naturally mimics the interaction of the mobility in the MLT with the database management system being distributed. The designing issues in storage of multimedia and its

nature of composition and decomposition is addressed in the mobile system. The OO approach enables the interoperability and expandability of the ASA.

Finally, the results analysis shows that the temporal constraints of the media file are met for both uploading and downloading transactions.

7.2 Recommendations

Though the prototype has proven the applicability of an MA-based architecture for managing the MLT of a DMMDB, it is the belief of this researcher that further study need to be done to improve and prune the architecture.

For future expansion the prototype can be modified by introducing more modules and/or upgrading existing ones in the ASA to support:

- **Logical partitioning and decomposition/composition**: the media module for decomposition and composition may be further studied and logical partitioning algorithms may be built.
- **Streaming**: another possible extension is also to consider online streaming than the offline streaming achieved in this work. Instead of a temporary file on a storage media, a data structure in memory could be considered for buffering applicable for buffering.
- **Better security**: better secured algorithms can be incorporated in the security module than the DES implementation tested on the prototype.

Though; three major aspects of improvement for future expansion are presented in here, more tuning can still be done on the proposed architecture. Further customization will make the architecture useful in multimedia applications such as: e-learning system, video conferencing, media collaborative work and interactive entertainment.

References

Adjeroh, Donald A., and Kingsley C. Nwosu. "Multimedia Database Management—Requirements and Issues." *IEEE Multimedia*, July-September 1997: 24-33.

Arora, Chetan, Paramjeet Nirankari, Hiranmay Ghosh, and Santanu Chaudhury. *Content Based Image Retrieval using Mobile Agents.* Delhi: Indian Institute of Technology, 1999.

Aygün, S., A. Yazici, and N. Arica. "Conceptual Data Modeling of Multimedia Database Applications." *Multi-Media Database Management Systems.* Dayton, Ohio: TUBITAK-BILTEN, Information Technologies and Electronics Institute, 1998. 182-189.

Bryce, Ciarán. "Security in the JavaSeal Mobile Agent System." Position Paper, 1998.

Dale, Jonathan, and David C. DeRoure. *A Mobile Agent Architecture for Distributed Information Management.* Southampton: University of Southampton, 1997.

de Groot, D.R.A., M.L. Boonk, F.M.T Brazier, and A. Oskamp. "Issues in a Mobile Agent-Based Multimedia Retrieval Scenario." *4th International Workshop on the Law and Electronic Agents.* 2005.

Deng, Lawrence Y., Timothy K. Shih, Teh-Sheng Huang, Yi-Chun Liao, Ying-Hong Wang, and Hui-Huang Hsu. "A Distributed Mobile Agent Framework for Maintaining Persistent Distance Education." *Journal of Inforamtion Science and Engineering 18*, 2002: 489-506.

Dunham, Margaret H., Abdelsalam Helal, and Santosh Balakrishnan. "A mobile transaction model that captures both the data and movement behavior." *Mobile Networks and Applications 2*, 1997: 149-162.

Eid, Mohamad, Hassen Artail, Ayman Kayssi, and Ali Chehab. "Trends in Mobile Agent ." *Journal of Research and Practice in Information Technology, Vol. 37, No. 4, November*, 2005: 351-359.

Hasse, Christof, and Gerhard Weikum. "A Performance Evaluation of Multi-Level Transaction Management." *17th International Conference on Very Large Data Bases.* Barcelona: Very Large Data Bases Endowment Inc., 1991. 55-66.

Jansen, Wayne, and Tom Karygiannis. *Mobile Agent Security.* Special Publication 800-19 NIST, Gaithersburg: National Institute of Standards and Technology, October 1999.

Katsumo, Yasuharu, Ken-ichi Murata, and Mario Tokro. *A Java Front End Approach for Programming Mobile Agents.* Tokyo: Sony, 2007.

Kothari, C. R. *Research Methodology - Methods and Techniques.* New Delhi: new Age International (P) Ltd., 2004.

Li, Weiyi, and David G. Messerschmitt. "Mobile Agent-based Network Signaling for Resource Negotiations." *Resource Allocation Problems in Multimedia Systems.* Washington: IEEE Real-Time Systems Symposium, December 1996.

Manvi, S. S., and P. Venkataram. "Agent-Based Subsystem for Multimedia Communications." 2006.

Milojicic, Dejan; Johansen, Dag; Kotz, Dave; Lange, Danny; Petrie, Charles; Rygaard, Chris;. "Trend Wars - Mobile Agents Applications." *IEEE Concurrency (July-September)*, 1999: 80-90.

Moser, Louise E. *Electronic Commerce Using Mobile Agent Technology.* California: University of California, 1999.

Piattini, Mario, and Oscar Díaz. *Advanced Database Technology and Design.* Norwood: Artech House, Inc., 2000.

Pleisch, Stefan, and André Schiper. *Approaches to Fault-Tolerant and Transactional Mobile Agent Execution – An Algorithmic View.* ACM Computing Surveys, Lausanne: École Polytechnique Fédérale de Lausanne (EPFL), 2004.

Ray, Indrakshi, Paul Ammann, and Suchil Jajodia. "A Semantic-Based Transaction Processing Model for Multilevel Transactions." *The Journal of Computer Security.* Oakland, 1998. 181-217.

Satoh, Ichiro. "A Hierarchical Model of Mobile Agents and Its Multimedia Applications." *Seventh International Conference on Parallel and Distributed Systems Workshops (ICPADS'00 Workshops)*. Iwate, 2000. 103-108.

Shih, Timothy K. *Distributed Multimedia Databases.* Taiwan: Idea Group Publishing, 2003.

Silberschatz, Abraham, Henary F. Korth, and S. Sudarshan. *Database System Concepts.* New York: McGraw-Hill Companies, Inc., 2002.

Tanenbaum, Andrew S., and Maarten van Steen. *Distributed Systems - Principles and Design*. New Jeresy: Prentice Hall, 2002.

Terziyan, Vagan, and Oleksiy Khriyenko. "Mobile Agent-Based Web Service Components in Semantic Web." *Eastern-European Journal of Enterprise Technologies, Vol. 2, No. 2*, 2004: 4-15.

Trappey, Amy F.C., Charles V. Trappey, Fiang-Liang Hou, and Bird F.G. Chen. "Mobile Technology and Application for Online Global Logistic Services." *Industrial Managment & Data Systems Volume 104 Number 2*, 2004: 169-183.

Wikipedia, the free encyclopedia - Mobile Agent. December 10, 2006. http://en.wikipedia.org/wiki/Mobile_agent (accessed September 28, 2007).

Yang, Yanyan, Omer F. Rana, David W. Walker, Christ Georgousopoulos, and Roy Williams. *Mobile Agent on the SARA Digital Library.* CACR Technical Report, California: California Institute of Technology, Center for Advanced Computing Research, 2000.

Appendices

Appendix A: Performance Test Result

Table 1: Downloading performance test result, without segmentation

		Download response time [sec]		Download transfer rate [KBps]	
Test No	File Size (KB)	[1 node]	[2 nodes]	[1 node]	[2 nodes]
1	275	0.66	0.64	414.16	429.69
2	648	1.04	0.84	623.68	767.77
3	834	1.16	0.99	716.49	846.70
4	1,041	1.31	1.02	797.70	1,024.61
5	1,450	1.49	1.20	971.85	1,205.32
6	2,116	1.92	1.47	1,100.94	1,440.44
7	3,077	2.42	1.88	1,270.70	1,641.07
8	3,961	3.13	2.20	1,264.28	1,798.00
9	5,085	3.93	2.67	1,294.06	1,903.07
10	5,998	4.59	3.02	1,305.62	1,988.73
11	7,485	5.73	3.53	1,305.26	2,119.80
12	9,012	6.63	4.13	1,358.66	2,184.73

Table 2: Uploading performance test result, without segmentation

		Upload turnaround time [sec]		Upload transfer rate [KBps]	
Test No	File Size (KB)	[1 node]	[2 nodes]	[1 node]	[2 nodes]
1	275	1.45	1.11	190.31	247.75
2	648	2.72	1.70	238.41	380.50
3	834	3.31	2.11	251.77	395.26
4	1,041	3.87	2.61	269.17	398.85
5	1,450	5.34	3.42	271.71	423.73
6	2,116	7.65	4.74	276.66	446.88
7	3,077	10.66	6.53	288.55	471.14
8	3,961	13.73	8.55	288.40	463.44
9	5,085	17.61	11.06	288.82	459.64
10	5,998	20.63	12.88	290.81	465.86
11	7,485	25.98	15.49	288.06	483.37
12	9,012	31.94	18.50	282.18	487.14

Table 3: Downloading performance test result, with segmentation

		Download average response time [sec]			Download total turnaround time [sec]			Download transfer rate [KBps]		
Test No	File Size (KB)	[1 node]	[2 nodes]	[3 nodes]	[1 node]	[2 nodes]	[3 nodes]	[1 node]	[2 nodes]	[3 nodes]
1	3,077	2.42	1.81	1.59	2.42	1.81	1.59	1,088.05	1,458.99	1,575.52
2	5,998	2.34	1.78	1.64	4.69	3.56	3.28	1,203.21	1,541.90	1,654.62
3	9,012	2.20	1.71	1.49	6.61	5.13	4.47	1,293.16	1,657.23	1,854.70
4	12,028	2.32	1.63	1.60	9.30	6.53	6.41	1,272.40	1,757.45	1,802.76
5	15,004	2.45	1.55	1.72	12.23	7.77	8.58	1,192.88	1,853.72	1,661.20
6	18,007	-	1.59	1.57	-	9.53	9.44	-	1,809.20	1,838.01
7	21,005	-	1.58	1.59	-	11.08	11.11	-	1,831.46	1,829.07
8	24,012	-	1.71	1.51	-	13.72	12.11	-	1,688.73	1,940.36
9	27,012	-	-	1.89	-	-	16.99	-	-	1,558.86
AVERAGE		**2.35**	**1.67**	**1.62**	**7.05**	**7.39**	**8.22**	**1,209.94**	**1,699.83**	**1,746.12**

Table 4: Uploading performance test result, with segmentation

		Upload average turnaround time [sec]			Upload total turnaround time [sec]			Upload transfer rate [KBps]		
Test No	File Size (KB)	[1 node]	[2 nodes]	[3 nodes]	[1 node]	[2 nodes]	[3 nodes]	[1 node]	[2 nodes]	[3 nodes]
1	3,077	10.56	6.92	5.53	10.56	6.92	5.53	1,088.05	1,458.99	1,575.52
2	5,998	10.36	6.84	5.38	20.72	13.69	10.77	1,203.21	1,541.90	1,654.62
3	9,012	10.39	6.54	5.40	31.17	19.61	16.19	1,293.16	1,657.23	1,854.70
4	12,028	10.46	6.36	5.28	41.83	25.44	21.13	1,272.40	1,757.45	1,802.76
5	15,004	10.62	6.33	5.25	53.11	31.67	26.25	1,192.88	1,853.72	1,661.20
6	18,007	-	6.75	5.18	-	40.52	31.09	-	1,809.20	1,838.01
7	21,005	-	6.55	5.07	-	45.84	35.52	-	1,831.46	1,829.07
8	24,012	-	6.59	5.09	-	52.69	40.73	-	1,688.73	1,940.36
9	27,012	-	-	5.25	-	-	47.27	-	-	1,558.86
AVERAGE		**10.48**	**6.61**	**5.27**	**31.48**	**29.55**	**26.05**	**285.38**	**447.09**	**564.38**

Appendix B: Comparative Test Result

Table 5: Comparative test result for video uploading

		Upload average turnaround time [sec]				Upload total turnaround time [sec]				Upload transfer rate [KBps]			
Test No	File Size (KB)	MA [1 node]	MA [2 nodes]	RPC [1 node]	RPC [2 nodes]	MA [1 node]	MA [2 nodes]	RPC [1 node]	RPC [2 nodes]	MA [1 node]	MA [2 nodes]	RPC [1 node]	RPC [2 nodes]
1	3,077	11.17	5.94	14.84	10.58	11.17	5.94	14.84	10.58	270.51	479.13	199.73	290.04
2	5,998	11.01	5.58	16.02	10.12	22.02	11.16	32.03	20.23	270.14	526.60	187.16	295.96
3	9,012	10.80	5.90	17.42	11.01	32.39	17.70	52.27	33.02	276.63	504.17	172.32	272.70
4	12,028	11.12	5.66	17.02	11.25	44.48	22.63	68.06	44.98	268.31	523.30	176.68	266.83
5	15,004	11.24	6.36	15.28	10.86	56.22	31.78	76.39	54.31	265.63	468.19	196.37	276.09
AVERAGE		**11.07**	**5.89**	**16.12**	**10.76**	**33.26**	**17.84**	**48.72**	**32.62**	**270.24**	**500.28**	**186.45**	**280.32**

Table 6: Comparative test result for video downloading

		Download average response time [sec]				Download total turnaround time [sec]				Download transfer rate [KBps]			
Test No	File Size (KB)	MA [1 node]	MA [2 nodes]	RPC [1 node]	RPC [2 nodes]	MA [1 node]	MA [2 nodes]	RPC [1 node]	RPC [2 nodes]	MA [1 node]	MA [2 nodes]	RPC [1 node]	RPC [2 nodes]
1	3,077	2.69	1.59	0.59	0.56	2.69	1.59	0.59	0.56	1,009.85	1,550.13	5,044.26	5,180.13
2	5,998	2.51	1.45	0.55	0.56	5.02	2.91	1.11	1.11	1,119.24	1,819.23	5,261.40	5,256.79
3	9,012	2.45	1.31	0.57	0.55	7.36	3.92	1.72	1.64	1,160.44	2,128.48	5,197.23	5,342.03
4	12,028	2.33	1.31	0.55	0.53	9.31	5.24	2.22	2.11	1,243.59	2,168.38	5,384.06	5,620.56
5	15,004	2.34	1.35	0.55	0.58	11.69	6.73	2.77	2.92	1,240.72	2,110.56	5,395.18	5,080.93
AVERAGE		**2.46**	**1.40**	**0.57**	**0.56**	**7.21**	**4.08**	**1.68**	**1.67**	**1,154.77**	**1,955.36**	**5,256.43**	**5,296.09**

Declaration

I, the undersigned, hereby declare that this thesis is my original work performed under the supervision of Prof. Sayed Nouh, has not been presented as a thesis for a degree program in any other university and all sources of materials used for the thesis are duly acknowledged.

Betiglu Mengistu

Department of Electrical and Computer Engineering,
Faculty of Technology, Addis Ababa University,
Addis Ababa,
October 2008