AN ADAPTIVE FRAMEWORK FOR COLLABORATION IN HETEROGENEOUS NETWORKS

by

PRAVIN BHANDARKAR

A thesis submitted to the

Graduate School-New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Master of Science

Graduate Program in Electrical and Computer Engineering

Written under the direction of

And approved by

_____

_____

_____

New Brunswick, New Jersey

October, 1999

ABSTRACT OF THE THESIS

An Adaptive Framework for Collaboration in Heterogeneous Networks.

By Pravin Bhandarkar

Thesis Director: Professor Manish Parashar

The ubiquity of network connectivity and recent advances in computing and networking technology have the potential of enabling computer-mediated information sharing and decision making in all facets of life including Medical Telediagnosis, Crisis Management, Electronic Trading. The ability to have direct and immediate access to all information defined by one's needs, interests and capabilities, will form the basis of each of these applications. However, enabling information sharing and collaboration in a networked environment where distributed clients can join, change their interests or leave at any time, presents many interesting challenges. Furthermore, client and network heterogeneity require information to be intelligently transformed so that it matches the client's local capabilities and resources, yet maintains semantic contents for effective sharing. This thesis presents the design, implementation and evaluation of an adaptive framework that enables seamless collaboration among distributed, wired and wireless clients, where the number of collaborating clients, their locations, capabilities and interests are dynamic. The framework is founded on an innovative semantic information coordination model that applies the "pull" knowledge management model to distributed information management. This is achieved by semantically enhancing messages and using state-based interaction techniques to communicate and replicate these messages in real-time. This thesis also identifies object-oriented design patterns that encapsulate information coordination and knowledge sharing, providing solutions to recurring challenges in developing distributed collaborative applications. An experimental evaluation of a Java based implementation of the framework is also presented.

Acknowledgements

I would like to thank my parents and my family for their love and support during my studies in graduate school. I am grateful to my advisor Professor Manish Parashar for invaluable guidance, encouragement and support throughout my stay at Rutgers. I am thankful to Professors James L. Flanagan and Ivan Marsic for their valuable advice and suggestions regarding my thesis. I wish to acknowledge the suggestions of the committee in developing my thinking, technical understanding, thesis writing and presentation skills.

I would also like to thank Bogdan Georgescu and Professor Peter Meer for developing the progressive image encoding and the image information transformation module. Interacting with other graduate students in the DISCIPLE project enabled me develop a better understanding of the topic and I would like to thank all of them for their support. Thanks to the CAIP computer facility staff, who helped with quick and detailed replies to various questions directed to help@caip. Lastly I would like to thank all the members of the TASSL lab for their excellent support and co-operation during the course of the project.

This work is sponsored in part by the NSF KDI grant (# IIS 98-72995) entitled “Multimodal Collaboration over Wired and Wireless Network” and CAIP Center. The CAIP Center is supported by the New Jersey Commission on Science and Technology and the Center’s Industrial Members.

Table of Contents

Table of Figures

Chapter 1

Introduction

Knowledge sharing and collaboration has always been critical to human activity. Human societies function through cooperation and teamwork. People exchange ideas, information and knowledge to achieve consensus-based decisions. The ubiquity of network connectivity and recent advances in computing technology has raised this interaction to a new level by introducing computer-mediated information sharing and consensus-based decision making in all facets of everyday life. Continuous advancement in technologies push the limits of the amount of information that can be processed and the rate at which it can be processed. Processing capacity of the chip is doubling every one and a half years (Moore's Law) and the DRAM capacity is increasing four times every three years[13]. This is complemented by advancements in networking which enable fast delivery of bits to the desktop. With the advent of technologies like dense wavelength division multiplexing(DWDM) networks are projected to reach the Tera bit/sec goal. Innovative techniques such as differentiated services [2] and integrated services [1] [40] promise required levels of quality of service on the universally prevalent Internet. Clearly the trend is moving towards providing quality time on networks at a reasonable price. These developments in technology in various areas together with the advances in network related technology provides an excellent basis for supporting collaboration in a variety of application areas. Application scenarios that are now feasible include:

Medical Telediagnosis: Paramedics rushing a patient from a distant location in an ambulance and collaborating with doctors at the hospital on the patient's E.K.G. The two locations are physically separated and the paramedics will have low end processing while the doctors at the hospital will have relatively high end processing ability.

Crisis Management: Natural disaster relief and civil emergency teams manage and deploy assets and contribute critical information (from the site of the disaster) via symbols on terrain maps.

Mobile Office: A person in transit connects with the office to participate in a videoconference and accomplishes office related work.

Electronic Trading: A broker can now virtually trade on the trading floor from remote locations during travel. A number of brokers can collaborate to discuss using real-time quote inputs and market inputs from brokers at more than one location.

The ability to have immediate and direct access to all information defined by ones needs, interests and capabilities, underlies each of the activities listed above. A design framework, which supports rapid information access and sharing across heterogeneous clients environments (wired and wireless networks, differing bandwidth, computing, and storage resources) will then serve universal access and increased human interaction. The objective of the thesis is to present the design of such a framework that enables seamless collaboration among dynamic groups of heterogeneous clients.

1.1 Collaboration in distributed heterogeneous environments

Collaboration can be defined as interaction and information interchange between people working at physically disparate locations working in dynamic heterogeneous environments with the purpose of accomplishing mutually beneficial activity[31].

Consider an interaction between paramedics rushing a patient from a distant location in an ambulance and collaborating with doctors at the hospital on the patient's E.K.G. The two locations are physically separated and the paramedics are continuously in motion. They will typically have a low end processing capability and a low bandwidth, lossy wireless connection, where as the doctors at the hospital have a high end processing capability; thus there exists a heterogeneous collaboration environment. A specialist's opinion may be needed who will than

have to join the original collaboration session hence the system is dynamic.

Figure 1 shows a block diagram of a typical collaboration framework. The different layer are discussed below.

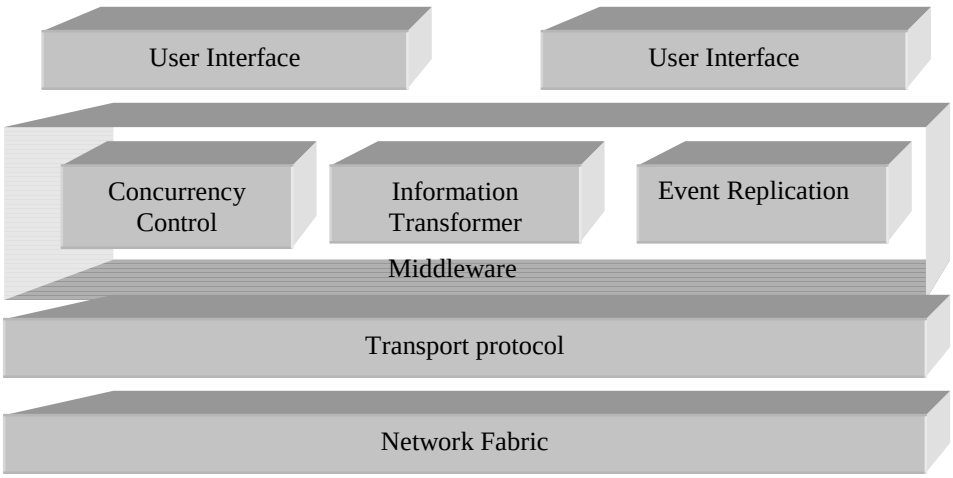


Figure 1 Layers in a typical collaboration session

1.1.1 Overall Architecture

The overall collaboration architecture defines the structure and operation of a collaboration session. Two schemes widely used schemes are

- centralized server based collaboration where control and management scheme for collaboration session provides a tight control on interaction within the framework.
- Distributed peer based collaboration system where the interaction between the peers is loosely coupled**.**

In a centralized server based scheme it is easier to maintain causality of events, concurrency control and regulate the events. However such a scheme is not scalable. It is therefore difficult to keep track of interests and capabilities of the various entities in such a collaborative session.

In a rapidly changing distributed system there is no notion of a centralized server to conduct the

collaborative session. The distributed peer interactions scheme would have a number of clients interacting at distributed locations. Such a scheme is scalable. It however would have to address issues of synchronization, concurrency and causality of events by loose coupling.

1.1.2 User Interface

User interface is a metaphor for the global virtual space in which all the clients of a collaborative session interact. It is imperative that the information exchange take place in a consistent manner across a widely varying group of clients. It should also be possible to incorporate new applications to this virtual space with minimum changes to the code.

1.1.3 Middleware

Middleware is an important layer of the collaboration framework. Key issues addressed by this layer are efficient event replication, concurrency control and heterogeneity management.

Event Replication: Event replication is the process of efficiently and transparently transmitting events generated by one clients action to all other clients in the collaboration session, and reproducing original action on the remote clients.

Concurrency Control: Concurrency Control is the process of arbitration and implementation of a control mechanism for maintaining consistent state when multiple clients manipulate the same set of shared objects in the collaboration session.

Heterogeneity Management: Heterogeneity management is the process of enabling effective interaction among a group of clients in environments with different capabilities and resources and performing necessary transformations to bring about consistent meaning to the shared space collaboration session. A requirement unique to heterogeneous networks is the interaction among a group of clients with varying interests and rapidly changing capabilities. This is especially true in interactions among clients on wired and wireless networks where the capability of the wireless network may change due to error rate or load on a wireless link.

Existing middleware architectures that can be used for collaboration include CORBA, DCOM and Java based architectures.

1.1.3.1 Common Object Request Broker Framework(CORBA)

CORBA is an architecture to enable seamless distributed computing [5]. The ORB is the middleware that establishes client-server relationships between objects. However CORBA was not designed with collaborative systems in mind. There are a number of changes that will have to be made in the current framework to adapt it to real time efficient replication. Some of the features that need to be addressed are:

- Base CORBA needs to be modularized to effectively eliminate the features that are not needed by real-time application. The dynamic invocation would incur too much overhead to be used by a real-time system.
- CORBA transparency allows invocation of an object from a remote node without any upper bound on the latency for remote-invocation when there is no control over factors like the network load. This can affect the performance of a collaborative system greatly.
- CORBA Event service defines the Event Channels as the broadcasters that forward all events from the suppliers to all consumers. The clients on the other hand may be interested only in a subset of events from the suppliers, and therefore they must implement their own filtering to discard unneeded events[36].
- CORBA does not provide any scheme to address heterogeneity management.

1.1.3.2 DCOM (Distributed Component Object Model)

DCOM[30] architecture from Microsoft has some interesting features that can help real-time collaboration. Some of the features are

- DCOM can identify the objects to which the client sends a request repeatedly and then can create such objects and inform the server. Upon request from the client for a

particular object, DCOM then activates the object on the server side. After initialization it needs only to send the id of the set to which the request is made and these requests could be sent along with other DCOM packets

- DCOM provides the ability to transmit "delta-requirements", a feature that can be used very effectively in real-time collaboration with respect to transmission of changes in events.

However DCOM has a number of problems that prove to be a major impediment for collaboration. DCOM lacks multi-platform support; the overhead of dynamic invocation can be substantial in DCOM. Also there is no mechanism to indicate the compliance to time guarantees.

1.1.3.3 Java Based Architectures

Initial evaluations of Java RMI[21] indicated that it would find it difficult to scale to multi-user sessions. It was also found to be slow for real time applications.

The info-bus specification from JavaSoft provides a model to enable collaboration. However info-bus enables interaction between the processes on the same JVM and does not support interaction between various JVM's on different machines. There are some proposed extensions to build bridges with other JVM's[16]. Until this is possible the info-bus cannot be used for collaboration.

1.1.4 Transport Layer

Transport level mechanisms form the basis of the communication architecture in a collaboration session. The underlying communication can leverage the Internet by the use of the Internet Protocol (IP) to exchange information. It is seen that with increasing network support for high bandwidths, there is a growing trend towards transferring images and voices on the network. High volume data to be sent to a group of users poses a unique problem for collaboration. Transport layer protocol such as TCP can be used for systems where the communication is point

to point between a small number of clients. TCP can ensure high reliability of data delivery but is not very highly scalable in an environment of dynamic clients. UDP based multicast schemes alleviate the limitations on scalability imposed by the TCP scheme. However reliability is not guaranteed. It is important to balance trade-off s of system requirements and the relative benefits of a collaboration system. Protocols like Real-Time Transport Protocol (RTP) [14] and Real Time Control Protocol (RTCP) together can enable transfer of real-time multimedia data. RTP is used to identify the synchronization source, transfer media data and sequencing support and synchronizing. RTCP identifies the participant, gets the data content, quality of service information, and requests retransmission.

1.2 Overview of the Thesis

This thesis presents the design of an adaptive framework to enable collaboration, and an extensible and effective implementation that forms basis to resolve collaboration issues. The framework includes an information management architecture, Semantic Information Management (SIM), Object oriented multi-layered architecture in terms of design patterns and a Java based implementation of the architecture. An experimental evaluation of the key aspects of the architecture is also provided. The key contributions of the thesis highlighted below.

1.2.1 Semantic Information Management (SIM) Model

SIM is an innovative interaction and information coordination methodology to enable distributed collaboration and knowledge sharing between heterogeneous (wired and wireless) clients. The fundamental innovation of the approach is the application of the “pull” knowledge management model to distributed information management. Events can be semantically enhanced using state-based [3] interaction techniques to communicate and replicate messages in real-time.

1.2.2 Object Oriented Architecture

This thesis provides an object oriented extensible architecture with a clear separation of

responsibilities. The architecture has been built with an emphasis on the separation of concerns relating to user generated events, information handling and processing, information transformation and processing by the communication media. Each layer addresses different aspects of the system; for example the application interface converts the event into semantic format while the communication layer deals exclusively with event transmission and reception.

1.2.3 Design Patterns for information coordination

The entire architecture has been abstracted in terms of design patterns that represents solutions to recurring software problems. These abstractions can be used in future implementation of collaboration architectures with approaches similar to the SIM architecture. This thesis proposes two new patterns to deal with information co-ordination and abstracting system state.

1.2.4 Heterogeneity Management

This thesis provides a framework for heterogeneous client environments to intelligently transform information so that it matches clients local capabilities and resources, yet maintains semantic contents for effective sharing**.**

1.2.5 Java based Implementation

This thesis presents a Java based implementation of the SIM architecture. This enables the framework to leverage inherent advantage of the Java delegation-event model hence events that are generated by the users can be captured by allocating event listeners to capture the events. The information coordination structure builds on Java based hash table structure to develop a multi-level hierarchical hash table. The multicast layer uses Java multicast for both data and event replication.

1.2.6 System State Abstraction

To make decision about application adaptation it is necessary to determine the state of the network. This thesis presents a network management module that uses the Simple Network

Management Protocol (SNMP) Parameters to determine the state of the network by querying the network elements [34]. This module uses two components: Java and WinSNMP API.

1.3 Outline of the thesis

In chapter 2 the various approaches and related work, to deal with event replication and collaboration are presented.

In chapter 3 fundamental abstractions to support the methodology and to be used as building blocks to realize such an information coordination and management infrastructure are identified.

In chapter 4 the interaction model is defined and a basic framework to enable collaboration in a heterogeneous environment among a group of dynamically changing clients with varying interests is proposed. These abstractions encapsulate system (network) behavior, quality of service, user mobility and the capability of the device to interpret information. The result is a *software architecture (or archetype)* for developing ubiquitous collaborative applications. A Java based implementation of this archetype is also presented.

Chapter 5 outlines the design patterns for distributed information co-ordination emerging from the usage of the semantic information management (SIM) architecture.

In chapter 6 an experimental evaluation of the Java based information sharing framework is presented. The SIM approach is contrasted with a point-to-point communication approach. SIM is also tested for adaptation to dynamically changing conditions.

In chapter 7 a summary is presented and the directions for future work are discussed.

Chapter 2

Related Work

Some of the notable research and work in the area related to this thesis have been described here with an overview of the architecture and messaging scheme.

2.1 Habanero (NCSA,UIUC)

The NCSA Habanero framework architecture uses the combination of a centralized server and an efficient messaging scheme to enable collaboration. It provides state and synchronization for multiple copies of the software tool. To determine the state of a remote computer, Habanero uses wrapping or “marshalling” of the present state of the computer using Java. Information is sent to collaborating computers using methods coded specifically to write and read the information. Variables that were assigned values through a graphical interface, or a computation, on a participant’s computer are shared with all of the computers in the session. The replication architecture works by sending sufficient information to each client to replicate the important state being shared by existing copies of that application. Habanero ensures in order delivery of events which results in applications appearing same to all the clients [4].

The Habanero architecture consists of a client and server to share Java objects. The server provides mechanisms for arbitration, routing and networking. Habanero works by replicating data and events in each client under the control of an arbitrator at a single server. The centralized server scheme is used to communicate among clients where each client application communicates to the server which in turn sends the events to the various clients in a session. The server controls the order in which events take place using a scheme of tickets that the serializer assigns each object. Arbitrators are central points of control to decide the order in which events are processed. The arbitrator code at the client ensures that the events take place in the order prescribed by the

tickets assigned by the serializer. The arbitrator has been extended from being a centralized one initially to a distributed arbitrator located at the various clients however the arbitrator at the clients is also regulated by the server based arbitrator. To keep track of the object Habanero uses a hierarchical naming scheme to ensure that events from a client are shared with the corresponding parts of other clients.

2.2 Tango(NPAC, Syracuse University)

The Tango system primarily aims at building a web based collaboratory[12]. It leverages its strength from the omnipresence of Java based technology that enables the applet to be obtained very easily using web browsers. This system is primarily a centralized server based system and the main functionality of the system consists of the following elements: session management, communication between collaborating applications, user authentication and authorization and event logging. Sessions have one distinguished user also known as the master of the session; he has special privileges of controlling the behavior of the application or other users accessing the application. The Tango system has two types of messages namely the control and application messages. Control messages are generated between the server, daemons(the process that run in the browsers and enable the applet to communicate with other clients) and control applications. These messages serve functions such as logging users into the system, establishing sessions etc. Application messages are means of communication between the user applications. The server logs events to record the system activity, since all the messages must go through the centralized server they can be recorded in the local database with the date, time and sender information.

To perform the communication between the applet and the central server there is a module which is implemented as a plug-in (live connect plugin for Netscape). This component maintains a two way communication between user applications, applets and the server, launching local applications, passing messages between applications running on the same node and providing functionality not normally available to Java applets. The centralized server routes all the

communication associated with the collaboratory. A unique feature of this project is the seamless collaboration between the application and applets.

Event replication in Tango allows the application to decide which application events need to be distributed. The manner in which events are distributed depends on the session management layer. Tango architecture enables scalability by not sending multimedia streams via central server instead supports multicast schemes for real-time data transfer. The usage of Live Connect Netscape Plug in to achieve message passing between the plugin and the applet, limits the portability of the system. Tango does not provide for object serialization and the user has to take care of marshalling and unmarshalling the object.

2.4 UNC Collaboration Bus (University of North Carolina at Chapel Hill)

The distributed collaborative framework at University of North Carolina uses an approach based on bus-agents. Bus agents are the bus contact points for clients of the bus. Through bus agents, application modules (1) register themselves and publish their connection characteristics to other modules, and (2) make connections to other modules, whether local or remote[17].In this framework a central server (a Java remote object), located at a well-known resource within a given Internet domain, maintains a central registry and allows users and other bus-agents to obtain remote references to active bus-agents. Bus-agents wishing to allow initiation of communication by another bus-agent or process, must register with the central registry.

2.5 Java Enabled Telecollaboration System (JETS) (University of Ottawa)

JETS is a collaboration system that downloads Java applets from a central server. Clients initiate a session by downloading Java applets from the server. The system maintains a repository of the application objects and replicates various client events on the application object. The server uses Java object serialization to send the state of the current object to the client. Centralized server is useful in keeping track of all the clients in a session and can relay the users to a newcomer or notify others when users leave the session[33][32].The communication between the

entities is performed by the server. The server multicasts events to various clients in the session. Management of a session is done by the centralized server as the server draws all the advantages of being the central node for communication; it also takes care of event collision and resolution of the events in case of event collision. This system leverages two important technologies namely Java based applets for collaboration and multicast for scalability. However it could be affected by the problems that typically affect a centralized system as the server may constitute a bottleneck.

2.6 DISCIPLE (CAIP, Rutgers University)

The disciple collaboration-enabling framework is based on a replicated architecture. In this architecture a user runs a copy of the collaboration client and each client contains a copy of the applet that is to be collaborated on. DISCIPLE uses Java Beans to build a component based architecture, the usage of this technology can enable ease of linkages with the other components. The idea of developing using JavaBeans is a preferred way to construct Java applications and components, since it supports visual programming and greater software reusability[35].

The exchange of messages between collaborating applications are of two types: 1. application state changes that need to be replicated are sent by multicast communication 2. Signaling messages to handle special situations are sent using unicast communication. Event adapters convert the arbitrary types of events into unified events that can be processed by the collaboration bus (C.BUS). The event object is converted into a byte stream using Java serialization. Communicator is the layer in the C.BUS, whose main purpose is to provide demultiplexing of requests as well as co-operative features such as concurrency, coupling etc. Communication is done by simple multicast protocol[15]. C.BUS automatically creates stubs and skeletons. The stubs are created based on receipt of remote object reference within the reply to a remote call. The event adapters on the other hand are created based on the receipt of a remote object reference within the reply to a remote call. Object references are obtained by using an object id.

2.7 Infospheres Project (CalTech)

The Infospheres project at CalTech aims at building a framework composed of two units, *dapplets* (which are multithreaded, communicating objects) and *sessions* (which groups of composed dapplets)[23]. A dapplet handles a message by possibly changing its state and sending messages on its output port.

Dapplets are composed together, *in parallel,* to form distributed sessions. A session is a temporary network of dapplets that carries out a task such as arranging a meeting time for a group of people. Sessions need not be static collections of dapplets; after initiations, their membership may grow and shrink. A session is specified in terms of a state transition: the state of component processes at the point in the computation at which the session is initiated and the corresponding states at the point at which the session terminates.

Each process has a set of inboxes and a set of outboxes. Inboxes and outboxes are message queues. A process can append a message to the tail of one of its outboxes, and it can remove the message at the head of one of its inboxes. Each inbox has a global address: the address of its dapplet and a local reference within the dapplet process.

Associated with each outbox is a set of inboxes to which the outbox is bound; there is a message channel from an outbox to each inbox to which it is bound. Each message channel is directed from exactly one outbox to exactly one inbox.

The distributed computing layer removes the message at the head of a nonempty outbox and sends a copy of the message along all channels connected to that outbox and also to the destination inbox of the channel. If the message is not delivered within a specified time, and exception is raised.

Associated with each session is the *initiator dapplet* that is responsible for linking the dapplets together. The initiator dapplet holds the addresses of all the participating dapplets and sends a request to all the dapplets to link up in a session. The dapplet may accept or reject the request

(because the requesting dapplet was not on the access control list, or because it is already participating in some other session.). A process, on receiving the initiated message binds its output queues to appropriate input queues and starts its thread, and thus begins its participation in the session.

After completing their tasks, the member processes close the session, having modified their local states. Indivisible resources are shared using tokens, which is either held by a dapplet or by the network of token managers. The dapplet initializes the token to a set value. To implement a simple read/write protocol with many dapplets, we use tokens such that a dapplet writes into an object only if it has all the tokens and reads from the object only if it has at least one token associated with the object. This system is distributed, and it coordinates multiple resource managers, each with its own policy. This could be very helpful in cases where more than one application or different sections of the same application need to be collaborated. The initial implementation of the system was using a UDP socket, with an intermediate layer to ensure reassembly in the correct order at the receiver.

2.8 Other Collaboration Schemes

In addition to the work cited above research and development in the area of collaboration is being conducted at MITRE corporation. The project Collaborative Virtual Work space (CVW) has built a tool to enable synchronous collaboration among a group of clients in the session. CVW enables virtual co-location through persistent virtual rooms, each incorporating people, information, and tools appropriate to a task, operation, or service [http://cvw.mitre.org]. From a user's perspective, CVW is a building that is divided into floors and rooms, where each room provides a context for communication and document sharing. Defining rooms as the basis for communication means that users are not required to set up sessions or know user locations; they need only enter a room. CVW allows people to communicate via text chat and audio/video

conferencing; import, store and retrieve data from virtual file cabinets; and mark up and write on shared whiteboards. All of this occurs within a persistent environment that provides the ability to retain continuity and provide true virtual co-location.

The primary objective of the lab space project at Argonne is to create virtual electronic laboratories that will allow distributed scientific research groups to collaborate naturally and effectively. The project has primary goals of satisfying the essential objectives of persistence, history, usage of existing tools, network independence, security, flexibility and scalability[37].

Chapter 3

Semantic Information Management (SIM) for Information Coordination

3.1 Semantic Information Management (SIM)Approach

A key requirement for real-time systems for collaboration and knowledge sharing for distributed (multimedia) applications (collaborative medicine, strategic battle planning and aerospace simulations) is the definition of an effective and efficient model for information coordination and replication in real-time. A further requirement for heterogeneous clients is the ability to locally interpret the events to reflect the interests, capabilities and resources of each client. Traditional distributed information management approaches are based on global naming services, where all communications use unique names assigned to clients. In such a system, every application client that enters a session must register itself with the naming server, explicitly stating its interests. The server then assigns capabilities to the entering clients and informs existing clients about the new client's interests. Existing clients can now forward relevant information from the existing collaboration session to the entering client. Clearly, the dynamics of such a collaborative framework is limited by the rate at which the network can synchronize distributing names, interests and capabilities. Furthermore, protocols for system reconfiguration, reorganization and the addition/deletion of clients are centralized at the server and can become exceedingly complex.

Semantic information management (SIM) [27] is a new approach for information coordination and replication to support real-time collaboration amongst heterogeneous, distributed and dynamic application clients. This approach implements the "pull" distributed interaction model using semantically enhanced events and state-based [3] communication techniques. In this scheme, each client maintains a profile that defines its current state, its interests and its

capabilities. All interactions in this scheme are then addressed to profiles rather than explicit names. As a result, the group of clients is determined only at run-time. In this formulation, clients can join or leave networks, and will result in truly distributed computation, liberating application clients from static sites and complex tracking protocols, allowing them to migrate freely and utilize available resources. Clients can leave a collaboration session by appropriately defining their profiles, without having to update complex membership rosters. State-based interaction techniques are the application of semantic content-based resolution techniques, used by the naming service, directly to the run-time interaction between clients. These techniques are naturally suited to multicast communication.

3.2 Semantic Information Management

The Semantic Information Management (SIM) approach formulates all interactions between dynamic sets of distributed, collaborating clients as state-based multicasts.

3.2.1 SIM Model

The overall approach is summarized in Figure 2. Each client in a collaboration session will locally export a "*profile*". A client's profile is a mutable set of attributes that specify its type, state, interests and capabilities. Profiles are maintained and modified by the clients to reflect their current interests. All communications between the collaborating clients are now defined as state-based multicast messages where a message is semantically enhanced to include a sender-specified "*semantic-selector*" in addition to the message body. The semantic-selector is a prepositional expression over all possible attributes and specifies the profile(s) of clients that are to receive the message. Thus the conventional notion of a static client or client group name is subsumed by the selector which descriptively names dynamic sets of clients of arbitrary cardinality (Conventional names of clients or clients groups are non-descriptive and statically bound.). State-based messages are received by semantically interpreting message selectors in terms of the client profiles. The interpretation is performed locally and asynchronously at the site

of each client.

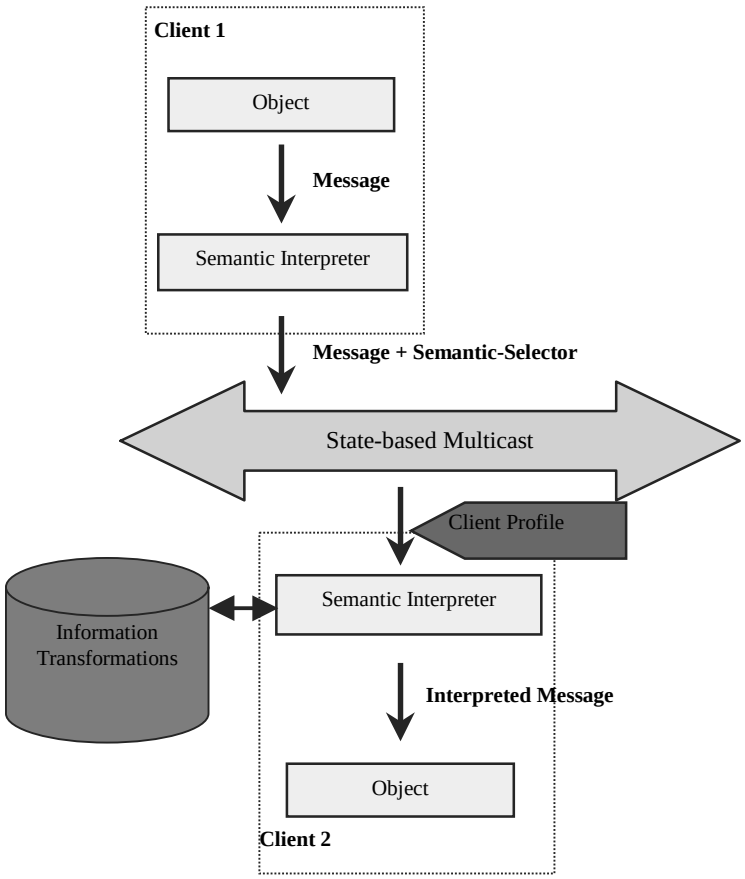


Figure 2 SIM Interaction Model

Figure 3 illustrates the semantic interpretation process. The semantic selector describes the attributes of the incoming stream as color video, with MPEG2 compression and 1 MB data.

Client 1's profile (Profile 1) matches this incoming selector and hence the message is accepted. Client 2 (Profile 2) on the other hand is only interested in B/W video with no encoding and so the message is rejected.

Client 3 (Profile 3) is interested in color video with JPEG encoding and has the capability to

transform MPEG2 to JPEG. It thus accepts the message with a transformation.

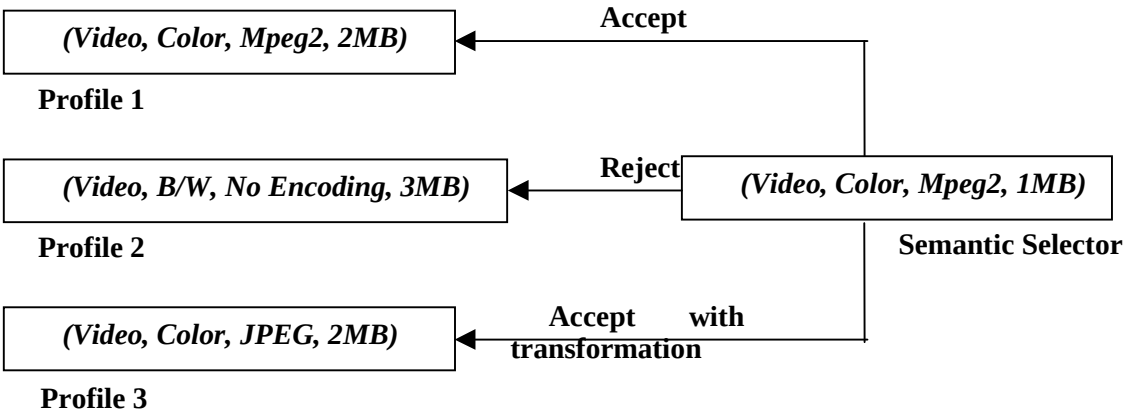


Figure 3 Semantic Interpretation

A related approach is used by the emerging Jini[22] technology for distributed resource sharing, which build on JavaSpaces[20].

3.3 Heterogeneity Management

The objective of heterogeneity management in collaboration is to enable the shared information to be intelligently transformed so that it matches the client's local capabilities and resources, and yet maintains semantic contents for effective sharing. To manage client heterogeneity in the SIM interaction model, semantic interpretation may involve information reduction and transformation, using information transformation modules, to meet the client's interests, resources and capabilities. For example, a client with a hearing disability can require speech information to be transformed into text. This can be achieved by appropriately setting the client's profile expression. Clients control their own profiles and can add, delete or modify attributes in the profile at any time. The modification of a profile is an atomic action with respect to its matching with a selector. The only global knowledge upon which such an interaction depends is the program-specific meanings of the attributes. This implies that all clients have information about possible application objects that can participate in a session and their states; information inherently known to all clients.

Chapter 4

An Adaptive Framework for Information Coordination

The objective of this section is to describe in detail the SIM architecture and identify the various abstractions needed to implement a software framework for information sharing and collaboration in heterogeneous distributed environments. The implementation of this framework is also presented. The key issues that need to be addressed by an adaptive collaboration framework are: event replication, event management, information transformation and system state interpretation.

4.1 Collaboration Framework

A schematic overview of the SIM-based collaboration architecture is shown in Figure 4. It

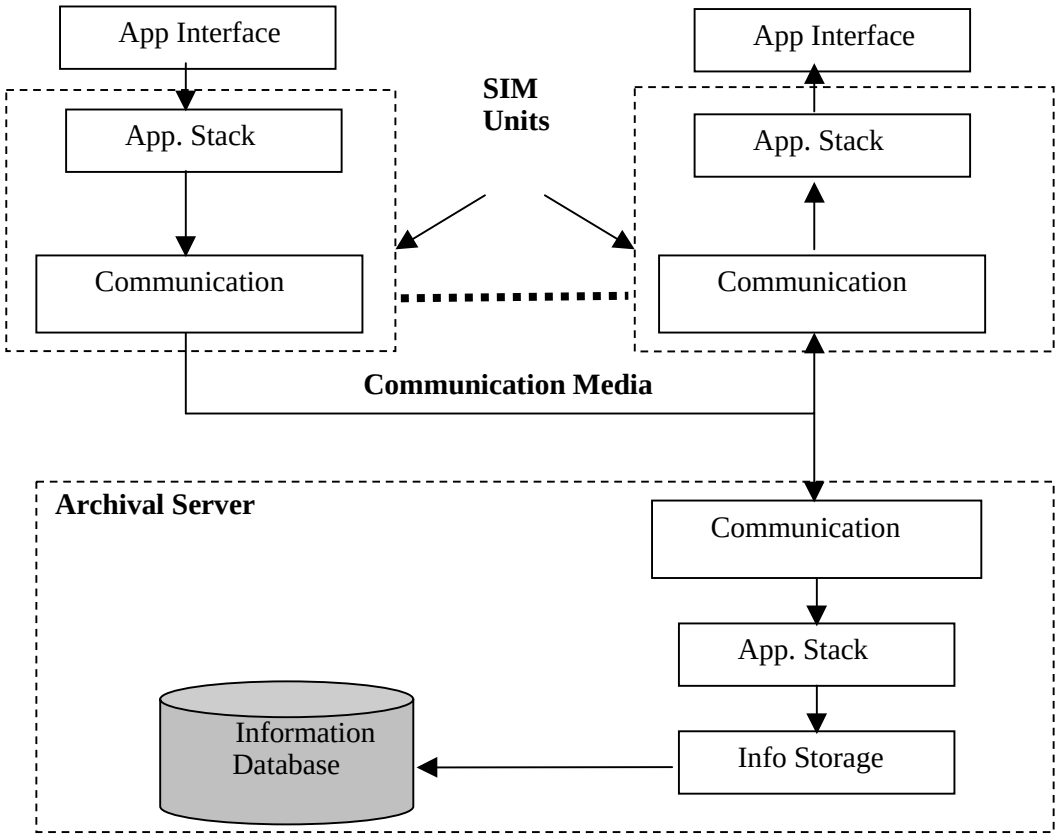


Figure 4 SIM Collaboration Framework: Architecture

consists of a dynamic number of application clients and an archival server. Each client site has a SIM-Unit (shown in Figure 5) which is responsible for transparently transmitting local events to all to all other clients, and receiving all remote events of interest to the local application client. The archival server maintains a repository of the objects in the collaboration session and their current states. The states are used by new application clients joining a session to obtain the current state of the ongoing session. It does not actively participate in a session. The different components of the architecture are described below.

4.1.1 User Interface

A clients local user interface (UI) represents a global virtual space where all the clients in the collaboration session interact. Locally generated events are transparently transformed into event in the virtual space and are visible to all clients interacting in this space. Similarly, remote events are transparently made visible to the local client. The UI is built on the SIM application interface.

4.1.2 Application Interface

The local application interface couples the user interface and the application stack. This component is responsible for locally orchestrating an application client’s collaboration session. It monitors all local objects of interest to the client and encodes their state as entries in the Globally Coordinated Object-State Table (GCOST). This update is then transparently propagated via the application stack to all "interested" clients in the session. Similarly, when a remote instance of the object changes state, the change is received by the semantic information interpreter(SII) and forwarded to the application interface (via GCOST) which in turn updates the client's session.

For example, consider a client session that contains a drawing canvas object. The client's interest in this object and its current state is locally recorded by the application interface as an entry in the GCOST. Each time the client interacts with the canvas (e.g. draws on it or changes the background color), the SIM application interface updates the GCOST entry to reflect the

change. This GCOST entry update is then propagated using state-based multicast to all clients in the session, the clients interested in the canvas object, filter and accept the event. Similarly remote events of interest (e.g. remote client changes the background color of the canvas object) are picked up by the local SII component from the communication media and replicated on the local object. The implementation of the user and application interface are Java based and enable users to leverage the elaborate Java delegation-event model[25]. The Java delegation event model works by assigning event listeners to low level events like *mousemoved*, *mousedrag*[1] and so on. The information is transformed into semantic messages by the application interface to enable collaboration.

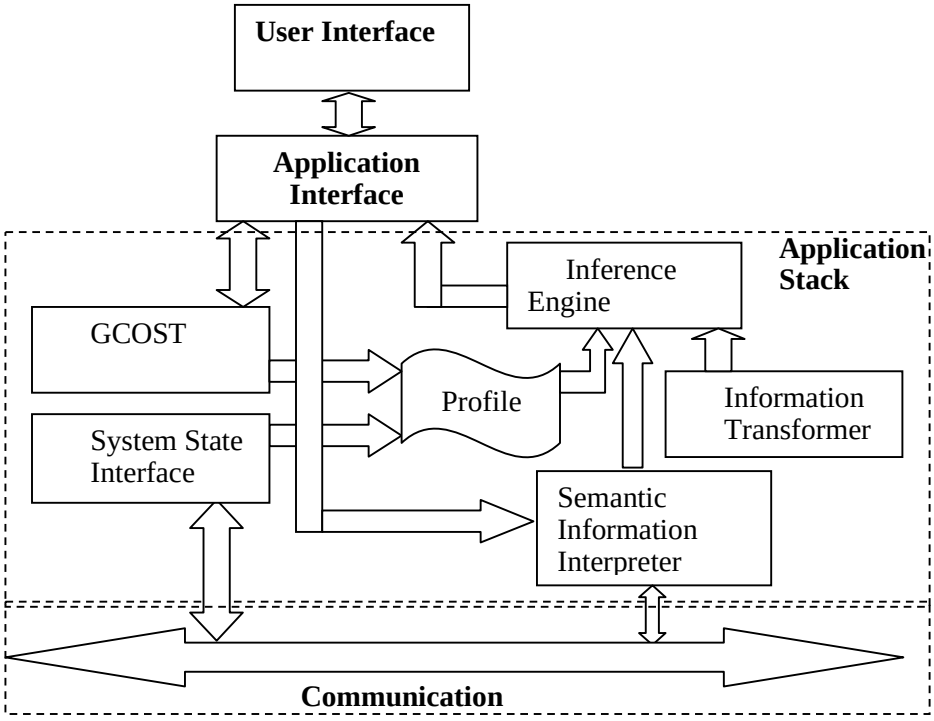


Figure 5 SIM Collaboration Framework: SIM Unit

4.1.3 Globally Coordinated Object State Table(GCOST)

The function of the GCOST abstraction is to maintain and coordinate the local state of the application. GOCST provides an associative query interface that is used to drive the local

[1] The functions in italics are implemented using the standard Java Interfaces available in JDK.

interpretation of information. The table is directly indexed, using the "semantic selector" from incoming messages, to determine whether the message is of interest to the clients, whether the client is capable of receiving the information, and what transformations (if any) have to be performed to match the information to the client's profile.

4.1.3.1 GCOST Design:

GCOST is structured as a hierarchical and extendible hash table. Each client maintains a local GCOST, and all local GCOSTs are globally coordinated so that if an entry exists in more than one GCOST, all its instances are consistent. The highest level of the hash table registers objects of interest to a client; subsequent levels maintain information about the current state, attributes and interests of each object. The lowest level of the table contains lists of "parameter – value" and "parameter – data" pairs. The storage at the lowest level is maintained in dynamically sized buckets, which are managed using extendible hashing mechanisms. (Extendible hashing[24]is a technique for efficiently managing dynamic databases by merging and splitting storage buckets.) The GCOST hash key space is hierarchically constructed by concatenating keys at each level. Searches into the table are performed by traversing the table hierarchy level by level. The most significant portion of a query index indexes into the highest level of the table identifying a particular object and subsequent portions identify object states. The least significant portion of the key corresponds to a particular attribute of the object. Each GCOST entry is a hierarchical

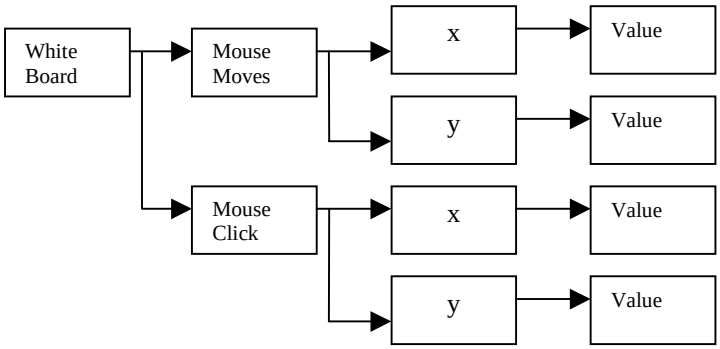


Figure 6 GCOST Entry

structure as shown in Figure 6 and corresponds to an object in the collaboration session capturing its current state and interests. The existence of a particular entry in the GCOST implies that the local client is interested in the associated object: the value at different levels of the entry defines its state and the events of interest.

4.1.3.2 GCOST Coordination:

An asynchronous coordination of GCOST entries is achieved using a "publisher – subscriber" interaction pattern built on state-based communications[29]. The message semantic-selector is uniquely generated using the GCOST hash key. The local communication process monitors message selectors and subscribes to messages when the semantically interpreted selector matches an entry in the GCOST. It then updates the local GCOST entry. This update is forwarded through the application interface to the user interface.

4.1.4 Semantic Information Interpreter (SII)

The semantic information interpreter is responsible for (a) associatively multicasting messages on the communication media, and (b) interpreting incoming messages, corresponding to remote events, for relevance and translating them into local events. A receiver component in the SII component monitors incoming event messages. For each incoming message, the SII extracts the semantic selector uses it to filter events that are interesting to local the client. This is done by translating it into the corresponding hash key and hierarchically querying the GCOST. The SII has two components namely the information receiver and the information sender.

4.1.4.1 Information Receiver

Application clients in a collaboration session are typically not interested in all objects in the session and hence should only process the relevant events. Furthermore, this set of relevant events changes over time especially as clients change their interests. For example, a client may iconize an object or hide it behind another object. In this case, events related to the iconized or hidden

object need not be processed. The SIM approach provides an efficient means for locally filtering such events without having to update central rosters. The information receiver uses the semantic header information to determine if the event has been locally generated or if event is remotely generated, obtains the profile from the GCOST to determine client interest in the remote event. If the event is relevant to current state of the object it is processed, else it is ignored.

4.1.4.2 Information Sender

The semantic event messages are transmitted over the communication media via multicast within and beyond a sub-net. The client transmits the event asynchronously to all the other interested clients in the collaboration session, without any knowledge of the current membership of this group or the interests or capabilities of other clients.

4.1.5 Inference Engine and Client Profile

The inference engine abstraction encodes policies for information transformations. It uses the application state (from the GCOST) and the system/network state (from the system abstraction unit) to generate the local clients profile. The former encodes the client's interests, preferences and capabilities, while the latter encodes available resources and their current state. The profile can be locally changed to reflect changes in the client or system state. The inference engine uses this profile along with the incoming semantic selector to determine the processing of incoming information[26].

4.1.6 Information Transformer

The Information Transformer maintains a suite of media-specific information abstraction algorithms. Information abstraction aims at intelligently reducing information content while maintaining semantics. Examples of information transformers include image-to-text, image-to-speech, text-to-speech, and speech-to-text conversions. Such a translation is critical for enabling collaboration across heterogeneous clients and interconnects with large variation in capabilities.

The information transformer library is designed to be extendible so that new modules and media types can be easily incorporated**.** The current implementation uses an image transformation module capable of *progressive description* of the visual data[9][18]. The module separates the minimal visual information essential to accomplishing the collaborative task. Robust segmentation of the image extracts a realistic sketch of the main features. This sketch, while preserving the essential information, requires 2000 times less data than the original. It can be translated to a verbal description and sent to wireless users to keep them in synchrony with other participants in a collaborative session. For users with more computational facilities, a more detailed version, at about 200 times compression can also be produced. The hierarchical representation of the same visual data, along with the verbal sketch, facilitates the collaboration among users in a heterogeneous network. Each of the users has access to the same visual objects but at different resolutions or modalities.

In the system, the inference engine uses the profile to decide on an acceptable resolution for the incoming that meets all requirements and constraints. The resolution threshold is used to determine the number of segments (i.e. the number of image packets) that will be received**.**

4.1.7 System State Abstraction

The system abstraction is a generic component that encapsulates the state of the system. This includes CPU load, memory available, network bandwidth, latency and jitter. In our implementation, this component is built using the Simple Network Management Protocol (SNMP) [34]to directly interact with network elements and obtain their state. It uses the IP address of the network element, community string and the object identifier (OID) of the parameters of interest (bandwidth, CPU load, page-faults, etc.). To obtain system state information from the local host we built a simple extension agent that runs on the local machine. This agent is built using Microsoft NT SNMP[19][34]. It executes as separate thread and

regularly polls instrumentation routines on the local host for system state parameters.

4.1.8 Archival Server

The archival server is a “passive” client in the collaboration architecture. Its sole purpose is to archive ongoing session as well as the current state of all objects in each session. This enables incoming clients to obtain the current state of an ongoing session. The archival server maintains its GCOST using the persistent storage. (Note that extendible hashing on which GCOST is built is primarily used for databases.) It passively listens to every published message and updates corresponding entries in the GCOST. Note that this server is only loosely synchronized with the clients and is only accessed when a new client joins a session. The archival server in our implementation additionally maintains a recent history of each session. It also provides a directory service of current clients in a session that can be queried only if a client needs to know this information.

4.2 Framework Operation

4.2.1 Overall Operation

Figure 7 summarizes the overall interactions between the various components in the SIM architecture. The events generated by the user interface are captured by the application interface, which appropriately updates the local GCOST table. This update is \encoded into a semantically enhanced message by the application interface and passed to the sender component of the SII that places it on the communication media. It is possible to add user-related information in the event. On receiving the data the local SII receiver can filter message based on their semantic selectors. Messages of local interest are accepted; other messages are rejected. Received event messages are used to update related entries in the GCOST. The inference engine is invoked to determine the interpretation of the incoming event and the required action is taken by the application interface. The application interface can replicate the events on the local user interface.

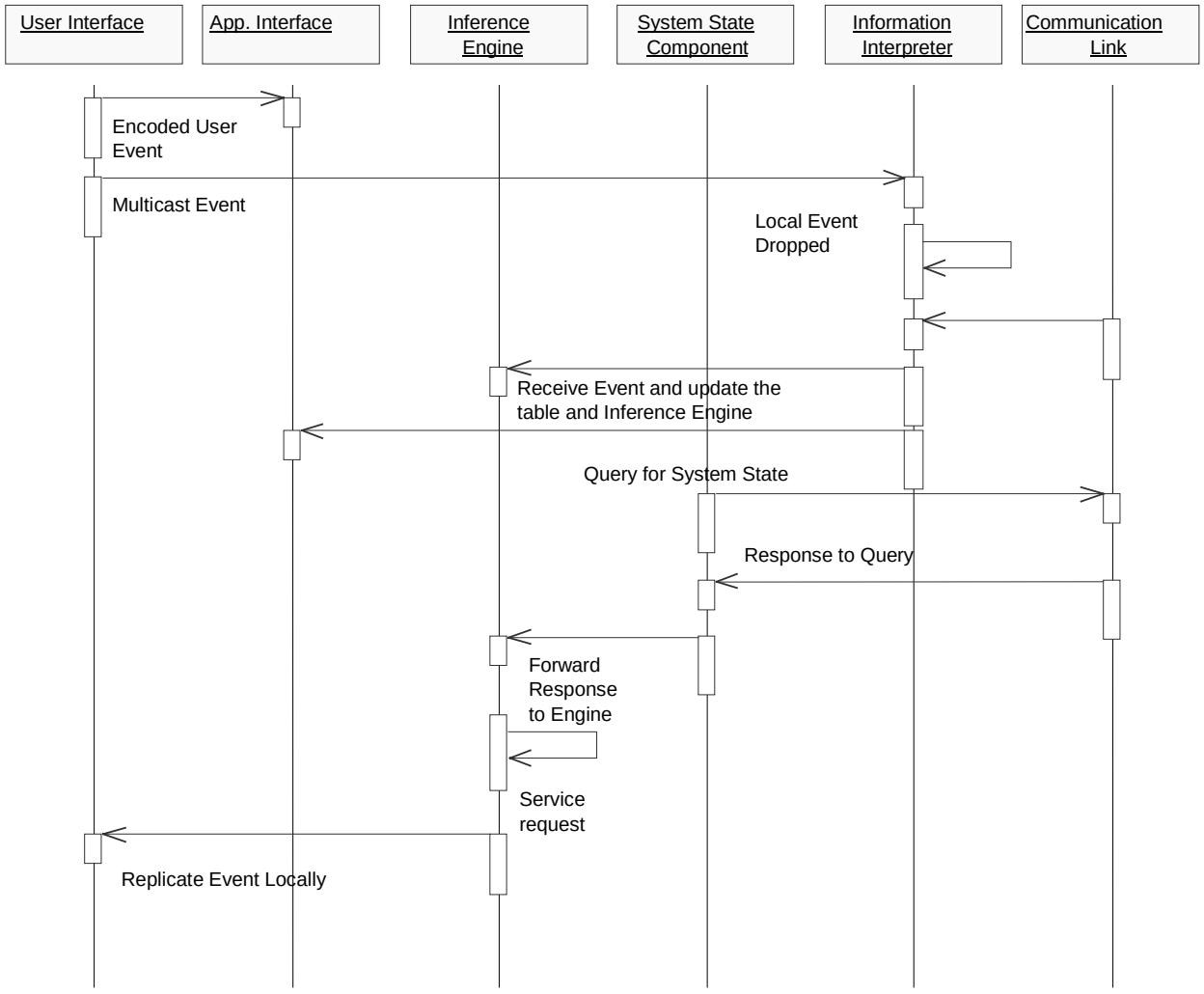


Figure 7 SIM Collaboration Framework: Operation

4.2.2 Concurrency Control of Events

Interaction between a number of clients at distributed location leads to the issue of concurrency control. Concurrency control in our implementation is administered via the passive archival server. As this server passively listens to every event in a session, it can detect concurrency violations. We use a simple detection algorithm using an event window; we consider event messages with timestamps within a certain tolerance as concurrent. When a conflict is detected, the server resolves it and subsequently sends out a resolved event. Messages from the

server are given highest priority at the client and are immediately processed. Another service provided by the server is to determine client priorities. Every client can negotiate a priority with the server. The priority is then added to the semantic selectors and can be used to resolve conflicts

4.3 Implementation of the SIM Framework

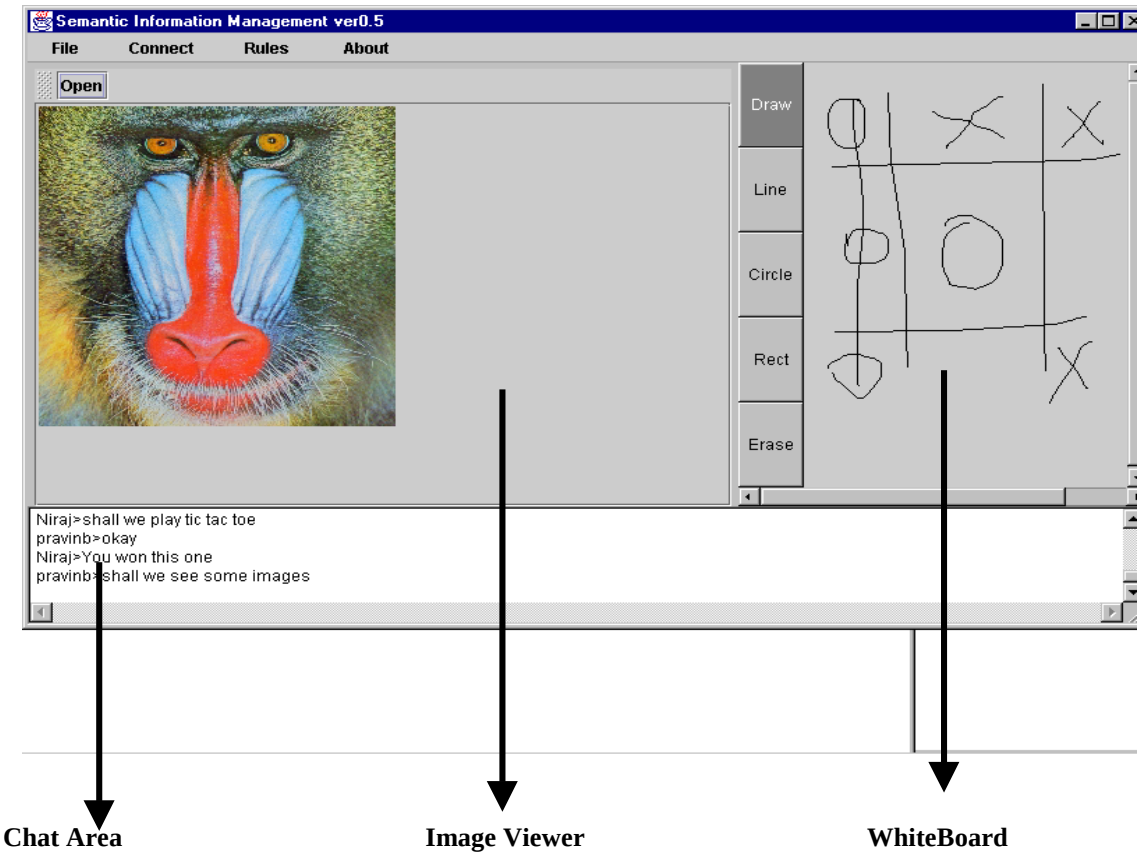


Figure 8 The SIM User Interface with the Various Components

The adaptive collaboration framework user interface using the SIM approach is shown in Figure 8. The framework is Java based and enables collaboration among wired and wireless clients. It incorporates three components chat area, whiteboard to draw shapes and an image viewer that uses multicast for all communication among the users in a collaboration session.

4.3.1 Multicast Communication

The SIM framework is built on a pair of multicast channels: the first is for control messages while

the second is used for image data. The main motivation for maintaining two separate channels is to prevent control messages from being blocked by larger image data messages. The separation of the multicast channels requires the users to subscribe to two channels to collaborate with each other. Upon logging in to the system the user starts collaborating by sending events and image viewer data, the SIM framework creates default channels for the transmission of events and data to the various users in collaborations session. If the multicast group is created then the default channel enables subscription to the channel. The SIM framework provides for the creation of new multicast channels before starting a new collaboration session, which are different from the default channels. The users could use the archival server to decide the multicast channels that they would use for the current session.

Multicast uses UDP packets for transmission of data, there is a need to ensure reliability in transmission of image data and events. The SIM framework ensures in order delivery of the image data packets. It has been seen that events require very small packets of data to be replicated on remote terminals, these data packets are delivered with minimum or no losses. The image data on the other hand is a high volume data and can entail delivery of many packets to be successfully reproduced on the remote terminals. The SIM framework uses a simple scheme that marks every outgoing image data packet with a number before transmission, this information along with the data payload is sent over the multicast channel to all users in the collaboration session. If there is packet drop owing to congestion at any particular router on the way then packets will be dropped. It is also possible for packets to arrive out of order, especially when the communication takes place over a wide area network. The packet numbers are stripped from the image data packets on reception at the remote terminal and verified locally to be in order. The transmission of the images is preceded by control messages to indicate that an image payload is being transmitted and the name of the image. Upon reception of these messages all the terminals in the session

ready themselves to receive the actual image payload. A counter is started and every image data packet received is initially checked the packet number before processing. The local counter keeps a track of the order of packets by incrementing itself every time a packet is received. If at any stage it is found that that packets are out of order, the first packet that arrives out of order is dropped and the decoding of the image is initiated. The image viewer enables the framework to view images with different number of packets. It is seen for real-time image data transfer that retransmission to ensure reliability can incur overhead and render the late reception of the data meaningless, hence we use better information transformation to keep the views of various users in the session synchronized.

4.3.2 Chat Area

Chat Area is built using Java AWT and the *keyListener*[2] interface. The chat functionality is integrated into the user interface and uses control multicast channel for communication. Keyboard events are trapped in the chat area and on construction of a sentence, the data is multicast to the various users.

Chat events are generated when a key is depressed by the user, these are trapped by the *keyListener*. The multicasting of each key event on the communication channel would put a processing overhead on all the users of the channel, furthermore as the number of users increases this communication will increase in direct proportion. This means that a substantial amount of processing and communication overhead would be incurred for sending a line of useful information. In addition to this if there are typographical errors on part of the users then each of the corrections will also have to be sent on the communication channel. To overcome this keyboard events are trapped and stored in a buffer till the local user is ready to send a line of data delimited by the " \n" key. Moreover corrections performed by users while typing a single line are performed atomically without a global notion of these errors nor the subsequent corrections.

[2] The characters in italics are standard interfaces, a part of JDK 1.2

4.3.3 WhiteBoard

The whiteboard is collaborative space in which users interact. It can also be used to collaboratively draw figures and schematic diagrams. The SIM whiteboard allows the user to selectively filtering events based on interests. For example if a user is currently interested only in lines, the software effectively filters out the other component events for example the rectangles, circles drawn by the other users. The implementation of the whiteboard board uses Java AWT components and AWT event listeners to trap and grab the events and replicate them on the various client location.

The whiteboard used was originally "unaware" application that was not collaboration ready. The "unaware" application is made collaborative by adding the thin layer of application interface that interfaces the whiteboard with the underlying application level stack to replicate events. The application generates low-level events for example *mousemoved*, *mouseclicked*, *mousedragged* etc. The application interface listens to the events and encodes them before transmission over the multicast channel. The application interface also replicates the events on the local user interface. The component architecture enables any whiteboard to be integrated into the framework. A important consideration in building the whiteboard was to integrate the whiteboard as a frame that could be easily replaced with another whiteboard component.

4.3.4 ImageViewer

The image viewer enables users to collaboratively view images. The image data is sent in an encoded format to all the users using a separate multicast data channel. When the local user views images on the local machine the data is multicast to other users in the collaboration session. The image viewer uses a progressive transmission for very low bit rate context based image compression. The image viewer also enables the user to encode images using compression algorithm and decode images with lesser number of image data packets (see Appendix A). More

details about the usage of the SIM software can be found on the web[28] .

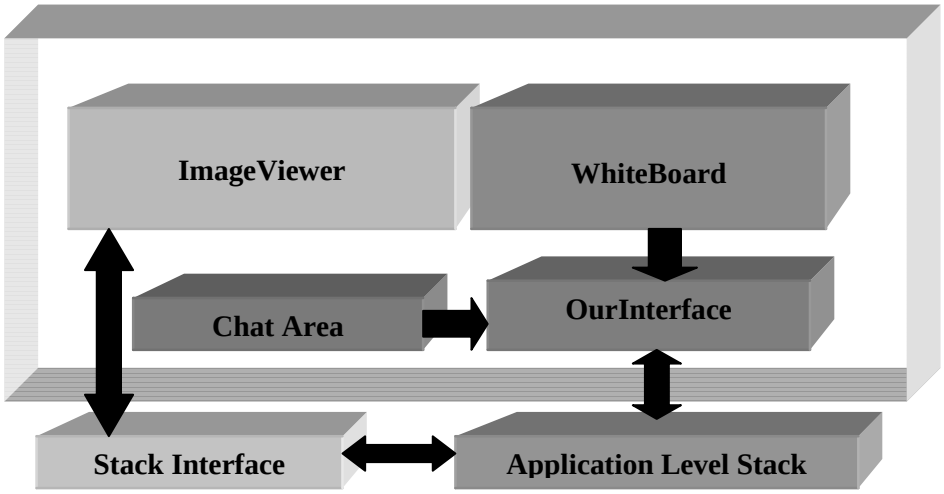


Figure 9 Interaction between the Various Components of the User Interface

The image viewer functionality is implemented using the *JImageView*, *ImageCanvas*, *ImageReader* classes.

The wavelet encoding software is implemented in C and is called using the JNI interfaces. The dynamically linked library for this implementation is bgcodec.dll of the various element of the user interface and their interaction with ***OurInterface*** classes. Figure 9 shows the organization of the various components of the Whiteboard and their interaction with the application level stack**.**

4.3.5 Application Interface Implementation

The Application interface component is used to convert the Java events generated in the user interface into semantically encoded information. The classes obtain the information related to the events from the components of the user classes. The application interface classes that perform the encoding to the semantic format are ***OurInterface*** and ***SEMWrite***r[2]. This information is then encoded into the semantic template after parsing the event obtained by the various listeners, implemented as interfaces. The objective of the application interface was to make the user interface collaborative by creating the instance of the class in the user interface class of the

collaborative software. The interface should be able to convert any collaboration unaware application easily into a collaboration aware application by invoking the instance of this class in the user interface where the listener interfaces are present. ***OurInterface*** class implements this functionality.

4.3.6 System State Component

The system state component is a generic component that can be used to determine the state of the system. The network state is a specific implementation of the generic component. It is implemented using network management to determine the state of the network elements and hosts. Since the software was evaluated on NT hosts, Windows based simple network management protocol (SNMP) was used to build the network management component of the software. There are two components in network management that have to work in conjunction with each other to enable acquisition of system related data. The two components are the manager component that runs on the management station and the agent component that runs on the network element to be monitored. To monitor the various network elements it is necessary to have agents running on them. Routers and switches have standard agents to monitor the local parameters through instrumentation routines. To monitor NT hosts on the other hand there is a need to build agents that run on the hosts and continuously obtain data network management data. This data is then forwarded to management stations upon request and used to monitor the behavior of the network element. The agents on the NT hosts have to be customized to obtain the data parameters as per the requirement and have to built separately. Such agents are known as extension agents.

4.4 Windows SNMP

Under Windows NT there are two SNMP services namely the agent service (***SNMP.exe***) and the SNMP trap service (***SNMPTRAP.exe)***. The SNMP agent service processes requests from the

[2] The class name in bold and italics represent the SIM framework implementation classes

SNMP management systems and sends GetReponse messages in the reply. The agent handles the interface with the Windows Socket API, SNMP message parsing and ASN.1 and BER encoding and decoding. The agent is responsible for sending the trap messages to SNMP management systems.

The SNMP trap service listens for the trap sent to the NT host and then passes the data to the management API. The SNMP service allows the user to build an extension NT agent that allows the MIB information to be dynamically added and supported as required. The extension agent resides within the SNMP service. It receives the SNMP messages across the network using the Winsock API, and passes the message data to one or more extension agents for processing.

4.4.1 Extension Agent using NT SNMP

Managed devices such as hosts and routers contain monitoring and (possibly) control instrumentation. The NT agent provides the instrumentation of some critical information such as CPU load for the manager. The NT agent represents hosts access to this instrumentation to the manager via a MIB, filtered by the SNMP security mechanisms. The manager communicates with the NT agent via SNMP to monitor and (possibly) control managed hosts.

The NT agent is based on the Microsoft SNMP Extension API that provides a basic functionality for constructing an extension agent dynamic link library (DLL) capable of communicating with the SNMP service and interacting with network management application using SNMP.

Once the SNMP service is activated on a host, the NT agent DLL is loaded as an extension agent DLL by the SNMP service. The DLL entry point function DllMain() is called first, then the initialization functions such as SnmpExtensionInit() and SnmpExtensionInitEx() are called to load the primarily supported MIB subtree, the handle used by the NT agent to assert that it needs to send the trap message, and additional MIB Subtrees if appropriate.

When a request message from a manager is received, then the querying process is invoked. Each request message will contain one or more variable bindings. The NT agent iterates through each binding and applies the Get, GetNext, and Set operation specified by the message type to the OID and the data value present in each binding. For processing each variable binding, the matching between the OID of the NT agent MIB variable and OID specified in the variable binding is checked first, then their attributes are compared. Finally actions will be taken if all the SNMP security checks have been passed.

4.4.2 WinSNMP based manager

The manager application is an SNMP-based network management application running under the Microsoft Windows NT4, and it is based on WinSNMP API[39]. WinSNMP provides a single interface to which application developers program.

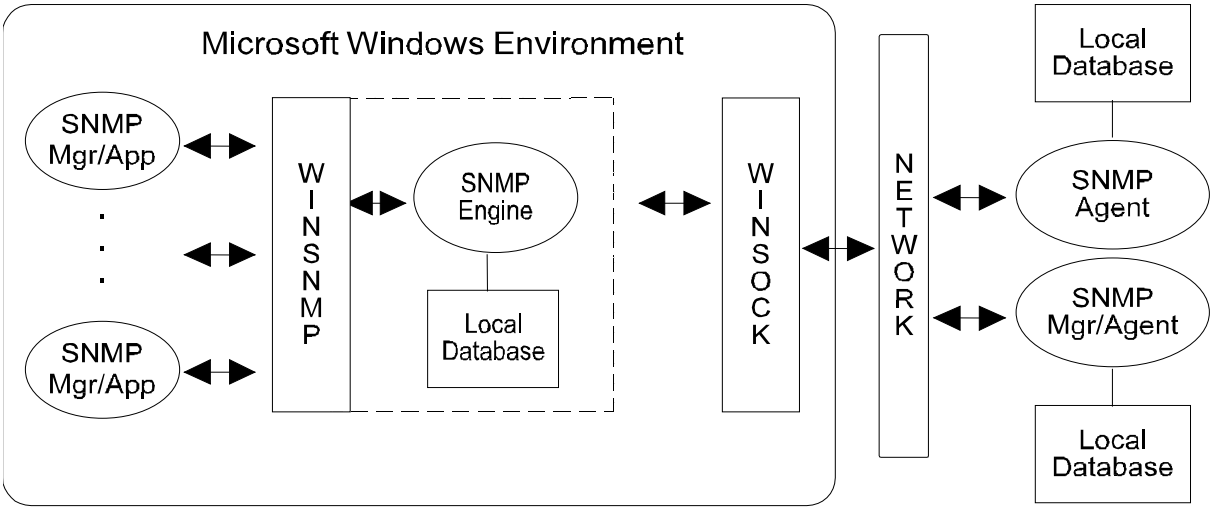


Figure 10 WinSNMP Architecture

Figure 10 shows where WinSNMP fits in one possible scenario of end-to-end SNMP connectivity from an entity acting in a managerial role (far left to an entity acting in an agent

role.). First a "session" is created by the *SnmpOpen* function that is used to manage the link between the WinSNMP application and the WinSNMP interface implementation. Then it uses *SnmpSendMsg* and subsequent *SnmpRecvMsg* calls to process querying information from the managed devices such as routers and hosts. Finally *SnmpClose* is used to close the session**.**

Chapter 5

Design Patterns for Distributed Information Coordination in Heterogeneous Environments

The objective of this section is to identify design patterns [10] that form an effective basis for creating a generic solution to address the various issues in collaboration and information sharing. The presented design patterns emerge from abstractions presented in the previous section and include proactive event acceptor, active event service pattern and system state pattern. The patterns are shown in Figure 11.

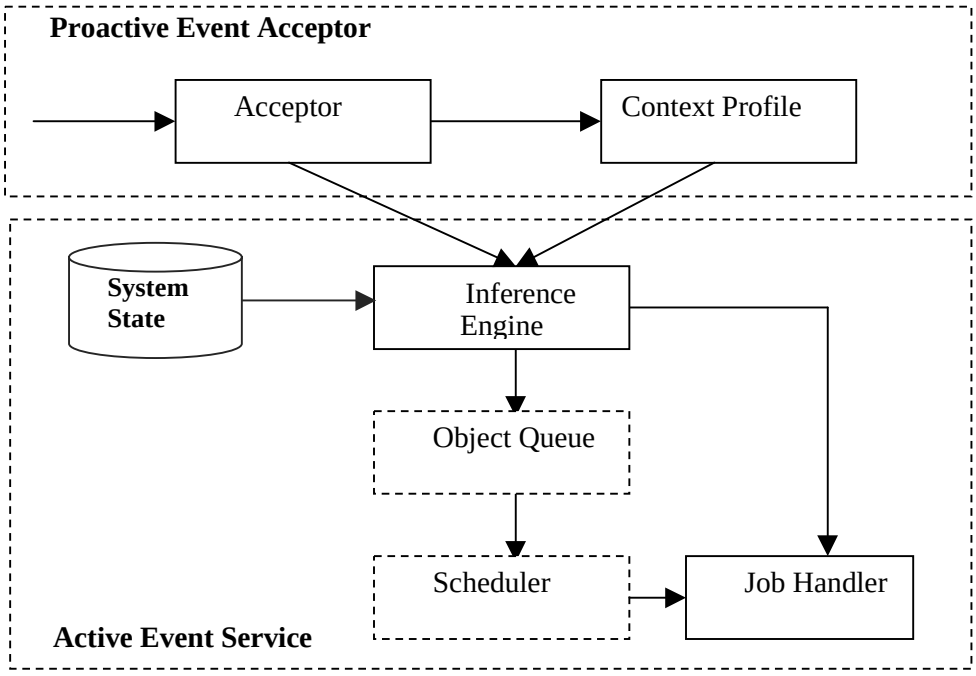


Figure 11 OO Design Pattern for Distributed Information Coordination

5.1 Proactive Event Acceptor Pattern

The *Proactive Event Acceptor Pattern* translates incoming events to client-specific events based on the local context. This pattern de-couples the acceptance of service events from their processing. The pattern has two components: the acceptor component that addresses the de-coupling of event reception and processing, and the context profile component that encapsulates

local context to determine the type of processing[6][7].

5.1.1 Design

In the implementation, the acceptor component runs in a thread and listens for semantic multicasts. The context profile is defined by the GCOST table, and reflects the current state, interests and capabilities of the client. The result is an adapted event that can then be processed based on the policy defined. The context profile is obtained by using the information from the GCOST table and used to make the context specific transformation as in the case of heterogeneous systems for example, converting an incoming event from text to voice to enable a hearing impaired user to collaborate.

The state table is the component used to maintain information related to the state of the objects in the collaboration session. The state acts as a repository of the information and can be queried to obtain information relating to events and determines if the events need to be processed or not based on the local client interests.

5.1.2 User Interface and Application Interface

The user interface is built in Java, enables the users leverage the delegation event model [25] to obtain the events generated. The Java interface provides with the low-level events which are semantically encoded by affixing the semantic header information and stored in the state table. The classes *OurInterface* and *SEMWriter* signify the application interface that are primarily responsible for converting the low level event into a semantic event.

SEMWriter
public static int object_count; public static MultiHash gcost_hash; public static OurInterface our_interface_to_owner;
public SEMWriter(OurInterfaceowner){our_interface_to_owner=owner;} public void create_gcost(String[ ] objects_collaborating,int objects){ **//create the state table**} public void event_update(String(event){**//parse the string obtained from the interface class and update the state table}** SEISender(event,gcost_hash,object_count,our_interface_to_owner); **//Send the event to the sender component of the message interpreter**

5.1.3 Proactive Event Acceptor Implementation

The state table forms the center of information storage and acts as a repository is the ***MultiHash*** class. This class is responsible for providing the context profile to process incoming events. There are two functions to get and set the profiles for the users. The acceptor portion of this pattern uses the ***SEISender*** and ***SEIReceiver***. The receiver runs in a thread and continuously listens for semantic multicast and picks up events from other clients in a collaboration session.

MultiHash
public void PutValue (String Level1Key, String Level2Key,String VectorKey,String parameter,String value) public String GetValue (String Level1Key, String Level2Key, String VectorKey,String parameter)

SEISender
DatagramPacket dp = new DatagramPacket(data,data.length,ia,port); thesocket.send(dp);

Together both the components accept and process user events and obtain local client profile to process the events.

5.2 Active Event Service Pattern

The *Active Event Service Pattern* extends the existing *Active Object Pattern* [7] to enable use of different service handlers based on local context (system state, application interests, type of information). This pattern implements context specific event processing where the event handling actively adapts based on local context and local policy. An important component of the pattern is the *Inference Engine* that encapsulates service policies. The inference engine determines which handler is to be invoked and accordingly schedules the job via the object queue**.** The inference

engine acts as a policy database to separate the policy and the servicing mechanism.

5.2.1 Design

The Active Event Service pattern extends and enhances the benefits obtained from the proactive active event acceptor. The main components of the active event service pattern are the inference engine that can decide on the basis of a rule base. The rule base can be set and modified by the user. The inference engine also schedules the job in an object queue. The object queue queues a job for processing by the service handler mechanism. The scheduler schedules the job to the event handler depending on the priority of the job. This allows the usage of different scheduling mechanisms depending on the user requirements. The object queue and the scheduler are optional components of this pattern and can be used when real-time events have to be serviced especially with varying priority[36].

5.2.2 Implementation

The code for this implementation is extensive and to explain the implementation and the flow of the program we have used pseudo-code. The events received by the proactive event service are forwarded to the inference engine only if the local client profiles reflect the client interest in the

```
Receive Event {
if (Process Event (using the  semantic template) == user interest)
                    {
                   forward to Inference Engine
                    }
else                 drop event
        }
Inference Engine code
Query state table
If( handler)
{
        Query various modules like network, local constraints etc
        Obtain all the values for the above parameters
        Compare values with rule base set by the user/ hard coded
        Process the event request based on the constraints
}
else drop event/ process with text message display.
```

event. The inference engine checks if there is a handler for the requests. It also obtains the network state related information from the system state abstraction. It compares the parameters with the user specified rule-base to decide on the best way to service the event.

5.3 System State Pattern

The *System State Pattern* maintains the local state of the system and defines the system context in which incoming events are serviced. Along with the context profile, it is used to drive the inference engine. For example, network abstraction is a specific instance of this pattern, which determines the current state of the network in terms of parameters like round trip delays, latencies and throughput. In our implementation, we use the information from the management information base (MIB)[8] that stores system parameters about the network elements such as switches and routers.

5.3.1 Design

The network management component is placed such that it obtains data on the state of the network. The abstraction can use various mechanisms to determine the state of the local network element or the local host and hence the network. This enables the component to place a set of constraints for the inference engine. The component queries the network and/or system elements to determine the state of the network as per the client needs. Network elements like the switches and the routers store information in a management information base (MIB). The MIB parameters enable the local component to determine information for e.g. Bandwidth can be found by determining the difference between the ifInOctets and the ifOutOctets at the local router interface. Additional information about the local host can also be found using the network management parameters at the host.

5.3.2 Implementation

The network abstraction is a simple class called ***snmp***, this class accepts the input rule-base

from the inference engine. The *snmp* class accepts the IP address of the network element to be queried, community string and the object id. This class interfaces with a dynamic linked library (DLL) ***javamgr.dll*** that holds the native code, which in turn interfaces winsnmp.dll[39]to query the network element. For the sake of the initial testing we used the CPU load parameter on an NT 4.0 machine to determine the time frame of reception of a particular stream.

SNMP

```
public long []_routerId = new long[11];
public String _router;
public String _community;
public long value;
```

```
public static native long snmpget(String router,String community,long[]routerId);
public snmp(String router,String community,String oid){ //Accepting the input in the
constructor}
static {
System.loadLibrary("javamgr"); // load native library}
public long getSnmpData(){
// code interfaces the c- native code to get the parameter
                              ……
return(value);
          }
```

To enable the acquisition of network management information from the local host we built a simple extension agent that runs on the local machine. The agent was built using microsoft NT SNMP[19]. The agent on the continuously polls the machine for the local data and upon request from the *SNMP* class, which is built on top of WinSNMP manager API, obtains the information from the instrumentation routines to reply to the SNMP get command. The user rule base was is the policy base for the network management parameters to be monitored. This rule base is entered by the user after logging on to the system. The test application was an image viewer with ability to encode and decode images based on wavelet transformation. The inference engine in response to an event to view images queries the network to interpret its state, then uses the parameter received to decide the time-frame to receive the image stream. This in turn reflects the response to the state of the network

5.4 Salient Points About Implementation

The Image viewer uses the encoding and decoding scheme for images is built in C and interacts with the Java code through JNI.

The SNMP querying component is built using WinSNMP dll and is loaded using JNI. This component queries the various network elements to obtain network parameters related information.

The interaction between the Java based code and the network management component can result in substantial timing overheads. It was found that if this information acquisition is done in the same module as data reception and servicing then it could result in an overhead on the real-time nature of servicing data information. Hence the SNMP module is made multithreaded and there are separate threads to service the different user requests for the network parameters.

The SNMP parameters to detect congestion in the network are at a coarser granularity and this could be made finer for example using the network management data from the RMON MIB to determine the state of the network. However the data obtained with finer granularity is liable to change very rapidly resulting in oscillations in the decisions made. Owing to this the SNMP parameters from the standard MIB's are used to make decisions about the state of the network.

Chapter 6

Experimental Evaluation of SIM

The experimental evaluation of the SIM collaboration framework consists of four experiments: the first experiment evaluates the framework in an actual collaboration scenario, while the second experiment uses a simulation of an extreme scenario, the third experiment on the other hand compares the behavior of the SIM architecture with a point to collaboration among clients in a dynamically changing environment and the fourth experiment evaluates the behavior of the SIM architecture in an environment of dynamically changing interests.

Parameters measured in the first three experiments were the setup time, client-to-client delay (CCD), and local client update time. Setup time included the time required to create and initialize the GCOST. Local update time was the time required for replicate the action locally once the event was received and included semantic interpretation, local GCOST update and user interface processing. The measured timing in both the experiments depended on the network load and the presented results are averages. In addition to network load, a significant overhead was due to processing in the underlying Java virtual machine, which depended on the number of events.

6.1 Actual Collaboration

Parameters Measured	Time in (ms)
GCOST table creation	3 – 5
Local client update	10
Client-Client Delay CCD	20-30

Table 1 - Experiment 1: Acual Collaboration

The actual collaboration session consisted of five Windows NT clients, including a wireless laptop, distributed across two different sub-nets. The session used three objects – chat area, white board and the adaptive image viewer. Each client randomly joined/left the collaboration session

and randomly defined its objects of interest. The measured parameters are summarized in Table 1. The CCD value was found to be well below the maximum of 60 ms for multimedia collaboration as prescribed by the Multimedia Communication Forum[33]. The effects clients leaving/joining the session were negligible. This may not be the case as we move to a larger number of clients and needs further evaluation.

6.2 Simulated Collaboration

In the simulated collaboration session, each client had a randomly varying number of objects of interest (from 0 to 6 objects) at each step of the simulation. Each object generated a random number (0 or more) of events. The average number of events that were generated in this simulation were of the order of 3000 per second, thus representing a stress test of the framework. The resulting measurements are summarized in Table 2.

Parameters Measured	Time in (ms)
GCOST table creation	3 - 5
Local client update	12
Client-Client Delay CCD	50

Table 2 - Experiment 2: Simulated Collaboration

The CCD parameter was again below the prescribed maximum for multimedia collaboration. Note that this experiment used broadcast to transmit the events both within and across sub-nets (instead of multicast used in the first experiment) which added to the CCD time.

6.3 Relative Comparison of SIM and Centralized Server Architecture

To validate the usage of the SIM architecture a test was performed with the set up used for simulation. A comparison was made between the behavior of a SIM system and a unicast point to point session of collaboration. The test involved the usage of the class built for simulation. The parameters monitored where the number of objects in a session and time for event replication. To demonstrate the dynamism of the system the number of objects were rapidly changing between 2-

6. The objects signified the user interest that was rapidly changing. The measurements for the normal collaboration and the SIM scheme were measured under similar network conditions so that the event replication times for each of the schemes could be measured.

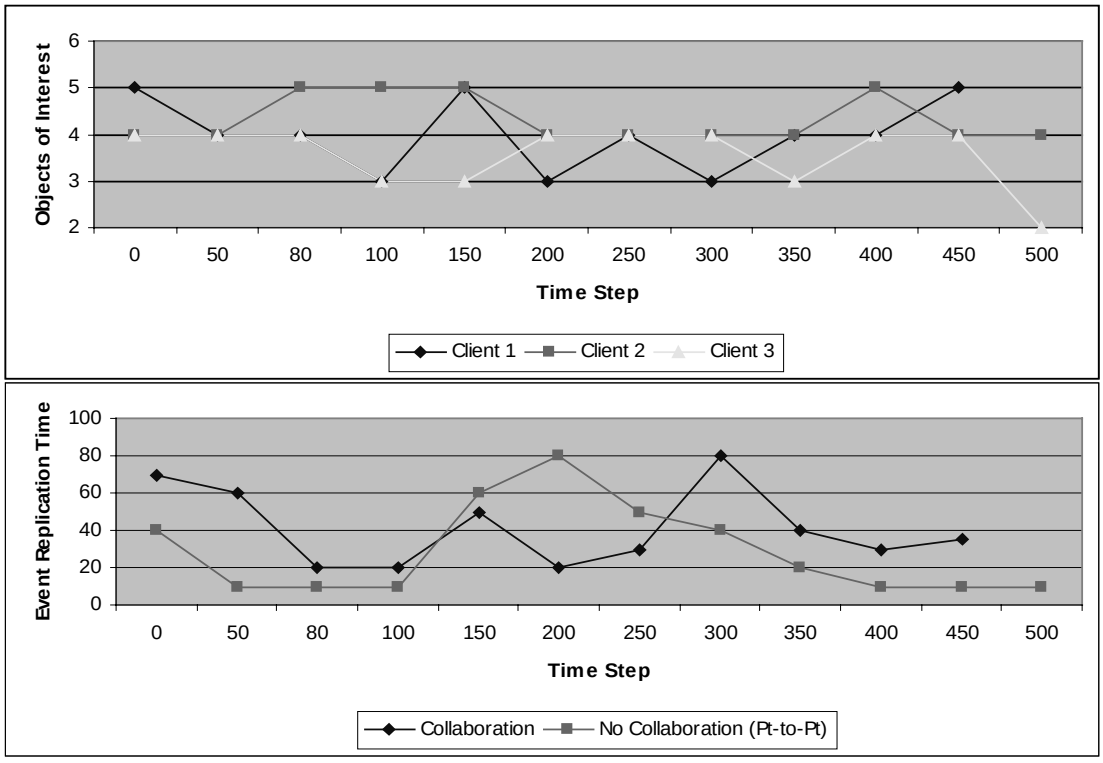


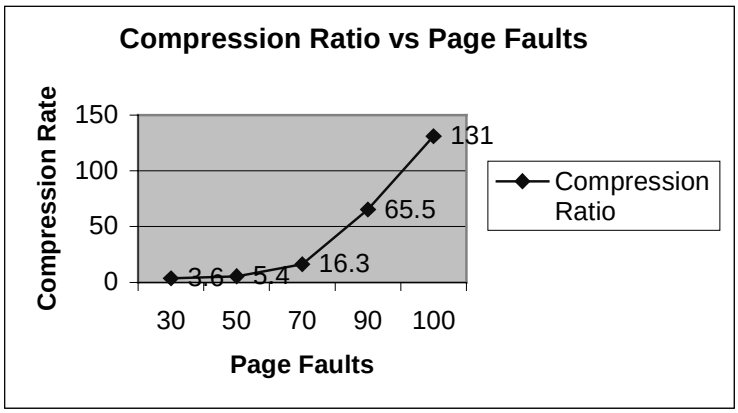
Figure 12 Relative Behavior of SIM and point-to-point scheme in an environment of dynamically changing interests

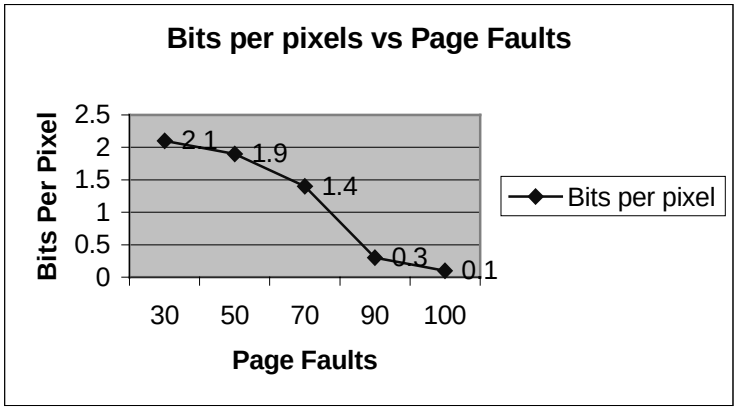
Figure 12 represents the result obtained from the test. From the first graph of the number of objects and the time step it can be seen that the number of objects are varying dynamically. The second graph traces the event replication time during dynamically changing conditions. It can be seen that that the event replication times for the SIM and the point to point architectures are similar. Thus it can be concluded that SIM introduces negligible overhead in event replication in a dynamically changing environment where clients with varying interests rapidly join and leave the collaboration session.

6.4 Behavior in Dynamically Changing Conditions

To demonstrate the adaptability of the SIM framework in dynamically changing conditions, SNMP parameters on the host and the image viewer performance were monitored. The experimental set up consisted of three NT work stations. The objective of the experiment was to determine the behavior of the image viewer software by plotting the compression ratios, the number of packets received and the quality of image (bits per pixel) versus the SNMP parameters obtained from the local hosts. The two SNMP parameters monitored were the number of page faults and the CPU Load.

6.4.1 The Image Viewer Parameters versus the Page Faults





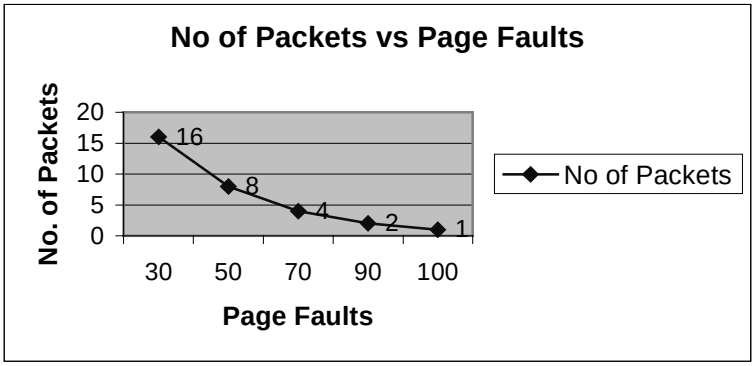


Figure 13 Three Graphs indicating the image viewer parameters versus page faults

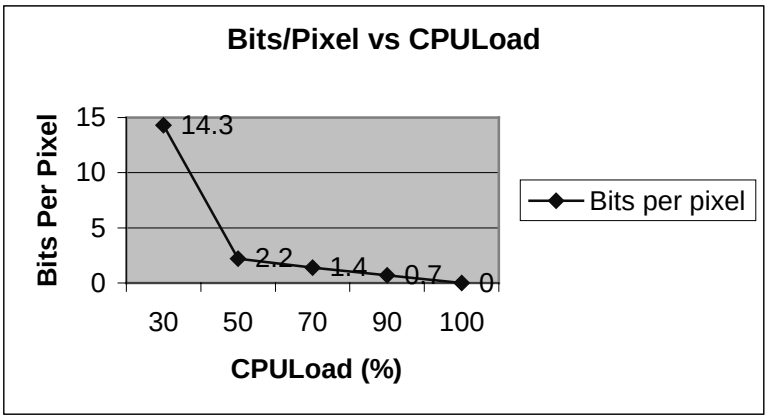
Figure 13 shows the adaptability of the SIM framework to changing conditions on the host. These measurements indicate that as the number of the page faults increases the number of packets that can be processed at the local host decreases.

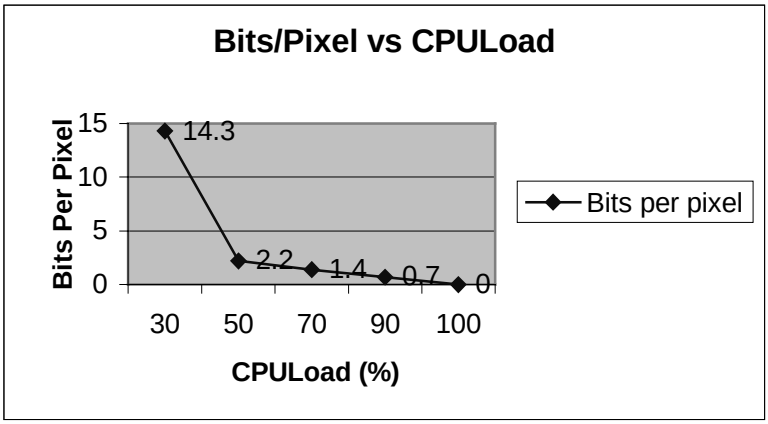
The number of packets vary from 1 to16 in powers of 2. This is set by the rule base. It can be seen that as the number of packets decreases the compression ratio increases indicating that lesser data is available for information transformation. The bits per pixel (BPP) is an indicator of the quality of the image, this parameter decreases as the number of packets decrease. The image viewer component adapts to provide images of varying BPP ranging between 2.1to 0.1 and compression ratios from 3.6 to 131. The number of page faults vary between 30-100 in steps of 20.

6.4.2 The Image Viewer Parameters versus the CPU Load

The compression ratio, bits per pixel (BPP) and the number of packets were measured for varying CPU Load conditions. Figure 14 shows the BPP, compression ratios of the images and the number of packets versus the CPU Load. The number of packets and the BPP decreases as the CPU load increases and the compression ratios increases in direct proportion to the CPU Load.

The variations in the BPP are between 14.3 and 0.7 for number of packets between 16 to 0.





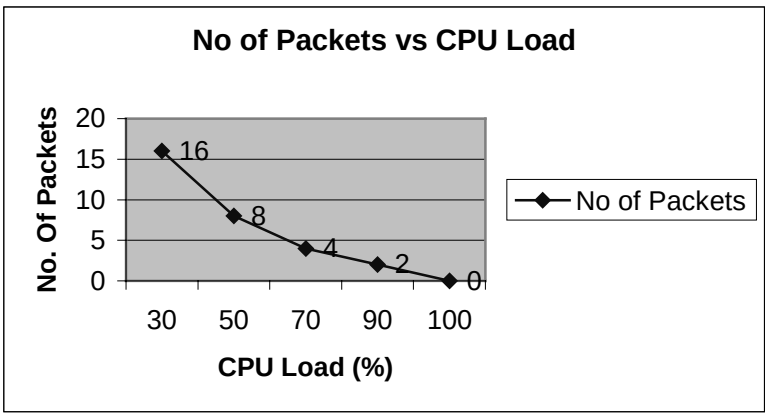


Figure 14 Three Graphs indicating the image viewer parameters versus CPU load

The image viewer can display images with varying compression ratios between 1.6 and 32.7.

From the above two experiments it can be clearly see that it possible to obtain a wide range of compression ratios and quality of images for different network/ system state parameters. This demonstrates the ability of the SIM architecture to obtain changing system state dynamically and adapt to varying conditions to transform varying data using the image viewer to maintain synchrony among heterogeneous clients.

Chapter 7

Conclusions and Future work

7.1 Summary and Conclusions

This thesis presented the design, implementation and evaluation of the Semantic Information Management (SIM), an adaptive framework for collaboration among distributed, heterogeneous (wired and wireless) clients. The framework is built on a semantic information model that implements the "pull" knowledge management model by defining semantically enhanced messages, and uses state-based interactions techniques to communicate and replicate these messages. This model introduces a different direction in the area of collaboration by addressing communication to profiles associated with individual clients rather than specific IP addresses. The information from the various clients is encoded in the semantic format and the header information for each in the form of a semantic selector is used to compare the incoming information with the client profile. The update of the client profile is performed locally and alleviates the need to update global rosters to indicate interests thus easing the mechanism of registering and de-registering interests at a centralized server, which can become a bottleneck. The usage of the SIM architecture is particularly suited to situations where the system \ client state changes rapidly.

The SIM architecture is built using object oriented design patterns for distributed information coordination in heterogeneous environments, and underlies. The presented architecture and design underlies a Java based collaboration framework. This thesis describes the Java implementation of the SIM architecture. The emphasis has been on the usage of simple a modular components and a lightweight architecture to enable efficient replication of events in real-time. The components enable user interface to be made collaboratively by adding an intermediate layer. Note that the collaboration is limited to low-level events, as there is no assumption of the

knowledge of the complex events.

7.2 Contributions

The main contributions of the SIM framework are: Semantic Information Management Object oriented design pattern based multi layer architecture that separates the functionality of the various elements, multicast communication for data and event replication and information transformation maintaining the semantic content. The implementation of the SIM framework to adapt to changing network conditions, distributed event handling, filtering, local interpretation based on local profiles and the semantic selector and includes usage of an image viewer module(Appendix A). The network management module to obtain SNMP information from the network elements and information transformation based on the information from the network management module. Current work is focussed on the evaluation of the framework with larger numbers of clients as well as the incorporation of other information transformation modules.

7.3 Future Work

The usage of distributed information co-ordination in heterogeneous environments introduces challenges in maintaining the causality of the events, event synchronization and concurrency control. The thrust of the future work will be in two directions: one would be enhancing the features in the current framework by adding different information transformation modules for heterogeneity management, enhancing the interpretation of the network and system state, adding new and advanced features to improve the user rule base. The rule base can form the basis of inference engine for decision making. The other thrust area would be the enhancement of the basic framework to incorporate the concurrency control mechanism, adding features to enhance the archival server and implementation of the concurrency and synchronization modules of the architecture outlined here.

References:

1. B. Bradeen et al. ,"Integrated Services in the Internet," RFC-1633, June 1994, www.ietf.org.
2. B. Carpenter et al .,"A Framework for Differentiated Services," Internet-Draft, March 1999.
3. B. Bayerdorffer, "Distributed Programming with Associative Broadcast," Proceedings of the 27th Annual Hawaii International Conference on System Sciences. Volume 2: Software Technology (HICSS94-2), Wailea, HW, USA, pp.353-362, 1994.
4. Chabert et al., "NCSA Habanero-Synchronous collaborative framework and environment," http://havefun.ncsa.uiuc.edu/habanero/Whitepapers/ecscw-habanero.html
5. Common Object Request Broker Architecture references http://www.omg.org
6. D. Schmidt et al.., "Proactor-An Object Behavioral Pattern for Demultiplxing and Dispatching Handlers for asynchronous events," Pattern languages for programming, September 1997.
7. D. Schmidt et al., "Active Object - An Object Behavioral Pattern for concurrent programming," September 1995.
8. Dave Perkins et al., "Understanding SNMP MIB's," Prentice Hall, Nov 1996.
9. E. Lamboray, "Progressive Transmission for Very low bit rate context based image compression," June 1997.
10. E. Gamma, Helm, et al., "Design Patterns," Addision Wesley, September 1994
11. Eliot Rusty Harold, "Java Network Programming," O'Reilly Pulishers.
12. Geoffrey Fox et al.., " Tango - A Collaborative Environment for the World Wide Web," NPAC Center Syracuse.
13. Hennesey & Patterson, "Computer Architecture: A Quantitative Approach ," Morgan Kaufmann

14. Henning Schulzrinne, " Real Time Transport Protocol," MCNC 2nd Packet Video Workshop, vol. 2, (Research Triangle Park, North Carolina), Dec. 1992.

15. I. Marsic, "A Collaboration-Enabling framework for Java -Beans," Rutgers University.

16. Info-Bus Specifications http://www.javasoft.com/beans/infobus/spec/index.html

17. J. Munson, "Collaboration Bus Infrastructure: Bus Agents," http://www.cs.unc.edu/~munson/DARPA/busagent.html

18.J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," IEEE Transactions on signal processing, San Fransico,CA, Pagers 657-660,1992.

19. James Murray, "Windows NT SNMP," O'Reilly publishers, January 1998.

20. JavaSpaces, http://www.sun.com/developers/hqbrief/javaspaces/

21. Jim Farley, " Java Distributed Computing," O'Reilly Publications.

22. Jini, http://www.sun.com/jini.

23. K. Mani Chandy, Joseph Kiniry, Adam Rifkin and Daniel Zimmerman, " A Framework for Structured Distributed Object Computing," California Institute of Technology.

24. Korth, Abraham Silberschatz, "Database Systems Concepts," 2nd edition ,1991.

25. Mary Campione, Kathy Walrath, "The Java Tutorial," http://java.sun.com/docs/books/tutorial/

26. P. Bhandarkar and Manish Parashar, " An Adaptive Framework for distributed Heterogeneous Collaboration,", to be published in Groups'99 Conference.

27. P. Bhandarkar, M. Parashar, "Semantic Communication for Distributed Information Co-ordination," IEEE conference on Information technology,1998.

28. Pravin Bhandarkar, "Semantic Information Management ver 0.5- User Manual," http://www.caip.rutgers.edu/~SEM/sim.html

29. R.Rajkumar, M.Gagliardi and L.Shua, "The Real-Time Publisher Subscriber Inter-Process Communication model for distributed Real-Time Systems," Technical report, SMI,Carneige Mellon University,1997.

30. Richard Grimes, "Professional DCOM programming," Wrox publishers.

31. Roseman, and Greenberg, S., "Groupkit: A Groupware Toolkit for Building Real-Time Conferencing Applications," In Proceedings of the ACM 1992 Conference on Computer Supported Cooperative Work (CSCW '92), 43-50, Toronto, Canada, November 1992.

32. S. Shervin, J.C. Oliveira and N.D. Georganas, "Applet-Based Telecollaboration: A Network-centric Approach," IEEE Multimedia, Spring/Summer 1998.

33. Shervin Shirmohammadi et al., " Java-Based Multimedia Collaboration: Approaches and Issues," University of Ottawa Canada.

34. SNMP Documentation, http://www.snmpinfo.com

35. Stephen Juth, "Collaboration Components for programming Real-time synchronous Groupware applications," Thesis report, CAIP 1999.

36. T-Harrison et al., "The Design and Performance of a Real Time CORBA Object Event Service," OOPSLA 1997

37. T.L.Disz et al., "Argonne's computing and communications infrastructure Futures laboratory"

38.W.Richard Stevens "TCP/IP Illustrated," chapter 26,Addision Wesley publication.

39.WinSnmp Documentation http://www.winsnmp.com

40.Zhang L et al., "Resource Reservation Protocol (RSVP)," RFC-2205,IETF.

Appendix A

(This work is contributed by Bogdan Georgescu and Peter Meer from Robust Image Understanding Lab (RIUL), CAIP Center)

The Image viewer is an image encoding/decoding system based on two channel coding. One channel is dedicated to the low frequencies, which carry average colors or gray levels of regions in the image. By delimiting regions in an image, the main contours are implicitly defined. The second channel is dedicated to the high frequencies, which carry the discontinuities like edges and textures. The low frequency information provides a low quality but accurate reproduction of the image. The details or the residuals for the whole image can be added through progressive transmission. This advantage of the software enables several users to share an image during a collaborative session. One user can send an image while at the receiving sites it can be optimally reconstructed given the available bit rate.

A.1 The Encoding Process

The image viewer software is based on the Embedded Zerotree Wavelet algorithm combined with run-length coding, DPCM coding of the lowest frequency components and arithmetic coding. The obtained bitstream is fully embedded and supports progressive transmission. Figure1 gives an overview of the implemented image encoder. The encoder accepts as input a color or grayscale image (PPM or PGM formats). If the input image is a color image, first a color space transform from RGB to YCrCb is performed and the chrominance is subsampled. The three resulting colorplanes are introduced as three separate grayscale image into the discrete wavelet transform coder (DWT). If the input image is a grayscale image, it is introduced into the DWT directly. The central part of the image coder is based upon the Embedded Zerotree Wavelet algorithm (EZW). The transform coefficients may be encoded in three different ways:

The coefficients of the lowest subband are encoded using DPCM and the remaining coefficients are encoded using EZW.

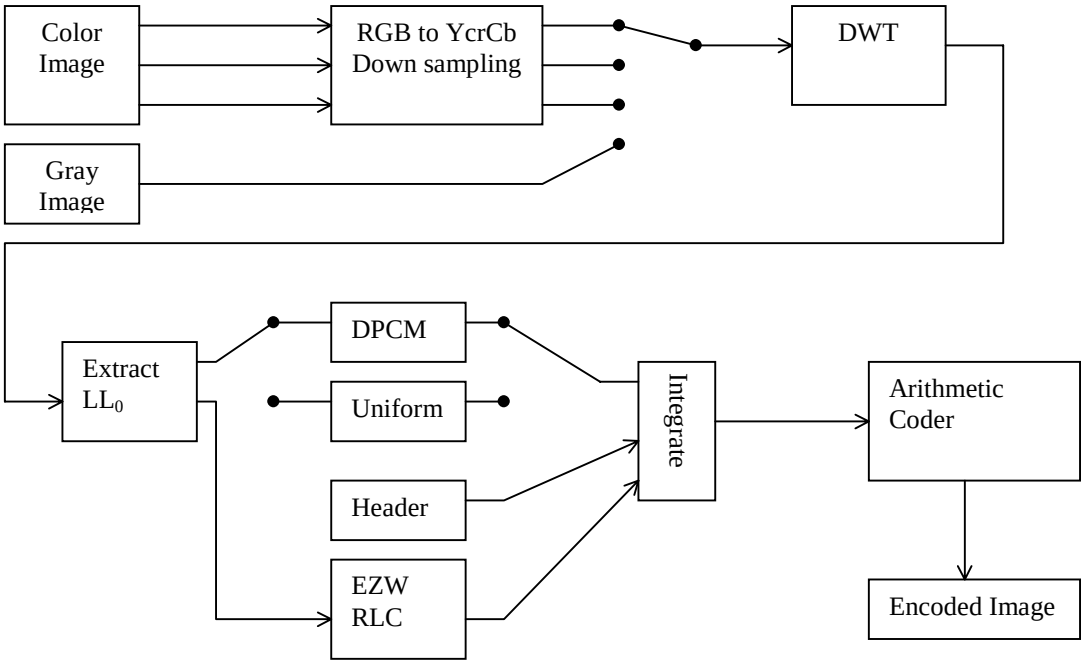


Figure 1 Image Viewer Encoding Process

The different modes cater to the requirements of different images for example if the input image is residual image resulting from the color image segmentation method then the combination of DPCM and EZW coding is better. If the image is not segmented then coding using EZW leads to better results. Along with the EZW encoding, run-length coding is performed (RLC). A header containing useful information for the decoder is built and all the resulting bitstreams are integrated and embedded. Finally the whole bitstream is arithmetically encoded.

A.2 Image Viewer Decoding

Figure 2 presents the decoding process. First the header is decoded in order to retrieve all the necessary information before starting to decode the image. The decompressed bitstream is decomposed into its components and, according to the encoding, the different subbands are decoded. An inverse discrete wavelet transform (IDWT) is performed and the output image is obtained. In case of color images the chrominance is first upsampled and the YCrCb bands are

transformed back into RGB.

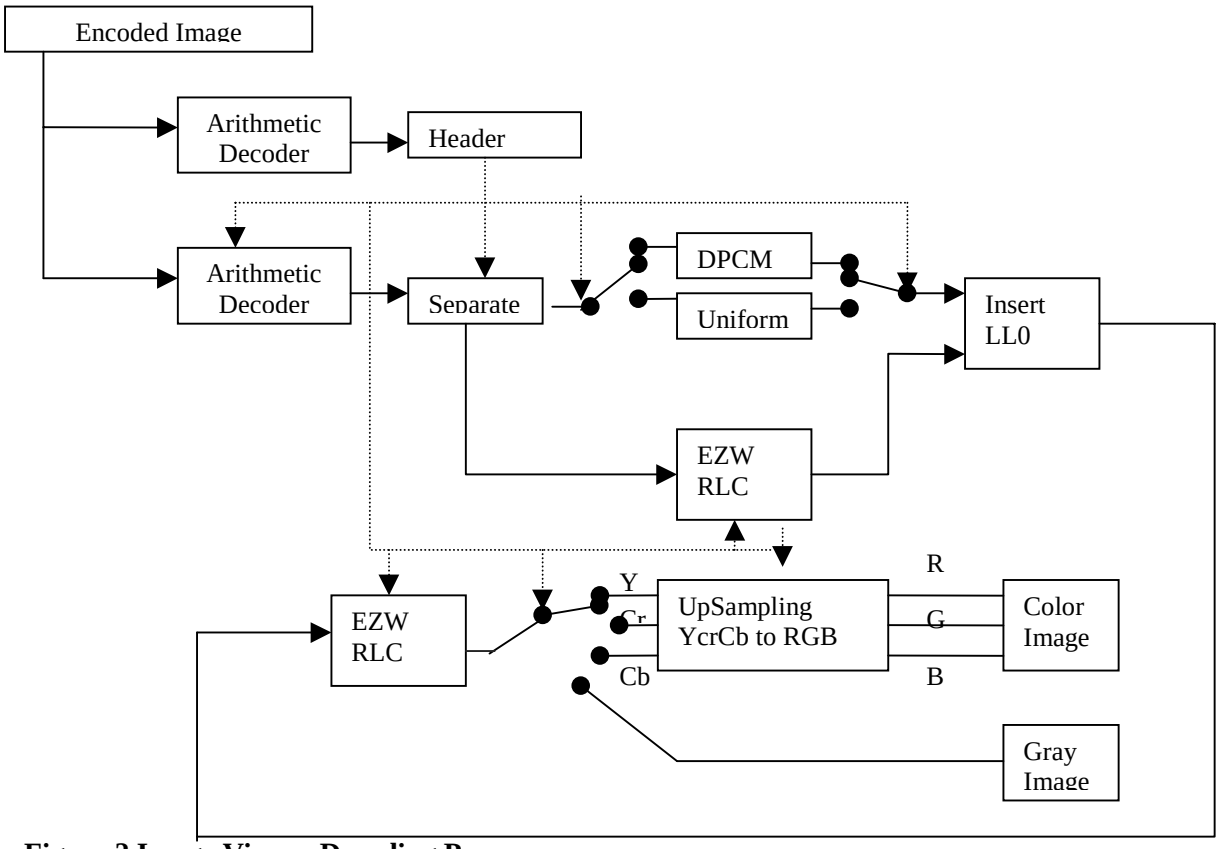


Figure 2 Image Viewer Decoding Process

The DWT is presented in an intuitive form, discussing its implementation using tree-structured filter banks. The signal is first split into its lowpass and highpass components. Each set of components is called a subband. Both subbands are downsampled by 2. The lowpass subband is recurrently split and down sampled using the same filter bank. We stop when the requested number of scales is obtained. Because of the recurrent splitting and dividing into lowpass and highpass components, these filters banks are also called octave-band filter banks. For reconstruction the obtained subbands are upsampled by 2, filtered by the respective inverse filters and added.

The results of the one-dimensional wavelet transform can be extended to the two-dimensional case, which applies for image compression. Considering the high pass filter H_h and the low pass

filter H_l, first we apply them to the rows of image X of size $2^n x 2^n$ and obtain two resulting matrixes H_lX and H_hX of size $2^{n-1} x 2^n$.

Then we apply the filters to the columns of H_lX and H_hX and obtain four resulting matrixes H_lH_lX, H_lH_hX, H_hH_lX and H_hH_hX of size $2^{n-1} x 2^{n-1}$. Each of these four matrixes represents a subband, H_lH_lX representing the average (low frequencies) of X and the other three representing the details. And we continue decomposition by further decomposing H_lH_lX. The transform coefficients can be interpreted such that at each coarser scale, one coefficient represents a larger spatial area of the original image but also a narrower band of frequencies.

The implementation based on the Embedded Zerotree Wavelet algorithm proposed by Shapiro is a simple image compression algorithm having the property that the bits in the stream are generated in order of importance. The algorithm is based on four key concepts: discrete wavelet transform or hierarchical subband decomposition, prediction of the absence of significant information across scales by exploiting the self-similarity inherent in images; entropy-coded successive-approximation quantization, universal lossless data compression through arithmetic encoding.