An Extensible and Secure Framework for Distributed Applications

Aneesha Sharma[1], Shilpi Gupta[2]

**[1]Department of Computer Science and Engineering, Amity University
Noida (201301), Uttar Pradesh, India**

**[2]Department of Computer Science and Engineering, Amity University
Noida (201301), Uttar Pradesh, India**

Abstract
Availability, Scalability, Reliability, Security and resource sharing are the key issues for success of any application, that are well addressed by distributed applications. Distributed applications provide services to different computers located at various locations that are connected by some means of communication network. In distributed systems a particular site consists of various computing facilities and an interface to local users and to a communication network. This paper provides various issues that must be taken into consideration while developing distributed systems. The issues discussed in this paper offer a secure framework for developing any distributed application on the top. Of these issues there are certain most commonly occurring issues that a distributed system fall victim to.

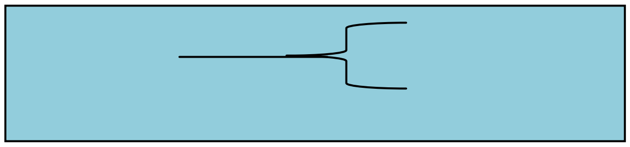
Keywords: *distributed system, message passing, consensus, clock Synchronization, deadlock detection, concurrency management.*

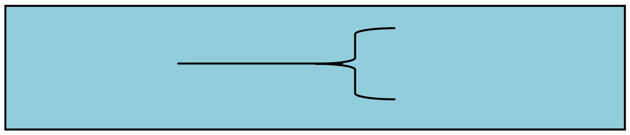
1. Introduction

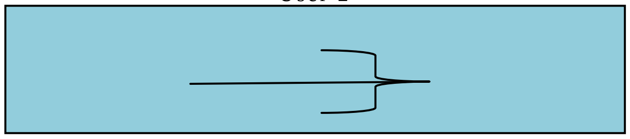
A Distributed System is viewed as a set of computers that are independent in nature and in which each computer has its own local memory, operating system and clock[1]. The means of communication in a distributed system is through message passing method. In message passing method there is a sender and a receiver. A sender communicates with the receiver by means of passing messages. The information to be shared is copied from the sender process's address space to the address space of all the receiver processes, and this is done by transmitting the data to be copied in the form of messages [2]. There are several issues that must be considered while developing a distributed system these include message passing, deadlocks, concurrency etc. All of these and many more will be discussed in detail in this paper.

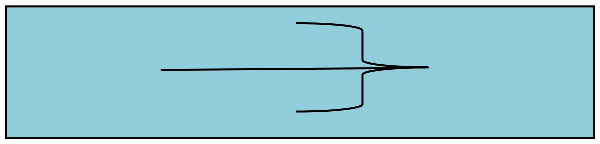
2. Problem Statement

As we know that distributed system is widely used in building many applications. So, through this paper an idea is given to build a Multi User Virtual Drawing Board (MUVDB) that uses the concept of distributed systems. Multi user virtual drawing board can support multiple users at the same time simultaneously and all the changes made by each user are reflected in all the virtual boards. The aim is to increase availability and provide scalability. Multi user virtual drawing board for extensible distributed framework is basically a drawing application that can support many users simultaneously so that all can work to achieve a common task considering views and ideas of all the users dedicated to perform the same task to design an application that is best in its design.

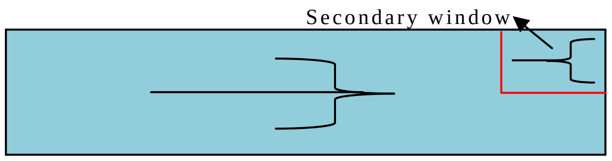

User 1


User 2

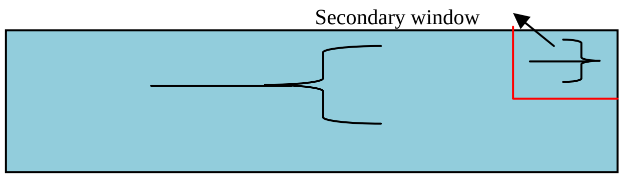

User 3



User 4



User 3 and 4 will have the same view



User 1 and 2 will have the same view

Fig. 1 interface design for MUVDB

The interface for the Multi User Virtual Drawing Board will look in the same manner as shown through the fig. 1. The users working on a design are free to give their respective views on a particular part of the design. The users that agree on the same design will be grouped together and a secondary window is also provided on the main window. This secondary window is provided in order to have the view of the designs made by other users. The number of secondary windows and groups can be controlled. The threshold can be decided by the number of users interested and supporting a particular design. The design instances can be discarded if no users are interested.The proposed idea of the MUVDB will be carried out in three parts. First a message passing model will be designed using sockets and threads. Secondly, an interface will be designed as shown in figure. 1 using swings. Lastly, concurrency or serialization protocol will applied. Through this paper only an idea for building such an application is given and no implementation is been provided for the same. The paper is divided into various sections starting from introduction, problem statement, basic architecture, issues in distributed systems, conclusion and future scope. Through the issues in distributed systems, an attempt is made to compare the proposed idea with the various issues been discussed in this paper. The aim is to state how these various issues can be handled in the multi user virtual drawing board and the basic advantage behind such an idea.

3. Basic Architecture

A distributed system consists of many computers located at different locations and all are connected via communication network. Each of these computers have their own local memory, operating system and clock apart from a Global clock. All of these clocks are synchronized in order to have effective error free communication. A machine at a particular site consists of two types of processes running on it. These processes are:

i. Local process (LP)
ii. Coordinator process (CP)

Communication between two machines in a network is carried out via Sockets. And communication between local process and a coordinator process is carried out via Inter Process Communication (IPC). Local process is basically used to draw an application and coordinator process helps the local process to see what is happening on other machines.

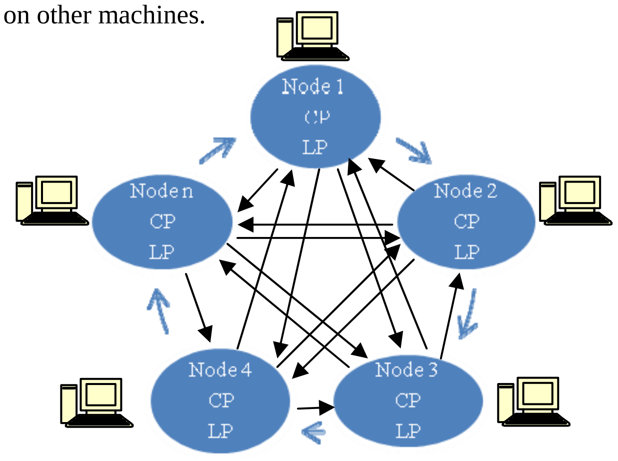


Fig. 2 Distributed Framework

4. Issues in Distributed Systems

Various issues in distributed systems are:

4.1 Message Passing Framework- In distributed systems there is a concept of inter process communication. There are two most important types of IPC's that are most commonly used these are shared memory model and message passing model. In both of these types of models failure may occur.

Conventional message passing technologies are:

4.1.1 Unreliable datagrams- identifies the corrupted messages among the stream of messages and then discards such messages. This technology generally fails because of its limitations to provide additional processing because of which most of the messages get through, some may get lost in transmission, duplicated or are delivered out of order [3].

4.1.2 Remote procedure call (RPC) - is said to be a reliable service. It works by providing communication by invoking a procedure that returns a result. But in this situation also failure may occur at the sender or at the receivers site. Failure may also occur in the network which may cause delay in the delivery of request or reply [3].

4.1.3. Reliable data streams- in this form of message passing technology communication is carried out over channels and these channels provide sequenced message delivery in a flow control and reliable manner [3].

ISIS is a system that provides various tools to support the construction of a reliable distributed system [3].

According to ISIS**,** Group communication involves various types of groups these are:

i. Peer groups- this type of scenario can be seen when the set of processes cooperate with each other, for example ,to replicate data [3].

ii. Client-server groups- in this type of group communication a process can communicate with any group provided the group name and permissions are given [3].

iii. Diffusion groups- are formed by a client-server group. In diffusion group clients register themselves and the members of the group send messages to the full client set and the clients are the passive sinks.

iv. Hierarchical groups- are built from multiple component groups, for the reason of scalability.

H. Attiya et al. [4] proposed a message passing model, in which various computations performed in the system are viewed as sequences of steps. In this model each step is of two types either a message delivery step delivering a message to the processor or a computation step of a single processor.

Message passing approach can be used in Multi User Virtual Drawing Board as the medium to exchange messages between number of users working on a common design and this can be achieved by building a message passing model using sockets and threads.

4.2 Clock Synchronization- distributed system is a collection of independent computers. The main reason behind the development of such a system is to achieve load balancing and resource sharing in the network. For this purpose it is necessary that the clocks of the communicating nodes should agree to a common clock value [1]. Now if the system is being employed to work for a real time application then it is must that all the clocks of the processors must match with Coordinated Universal Time, UTC.

Factors that cause errors in the clock are:

i. Clock skew- occurs when two clocks run at an exact same speed but have a constant difference.

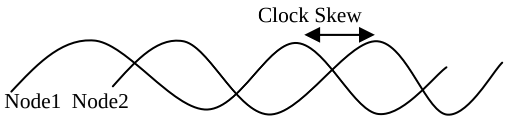


Fig. 3 Clock Skew

ii. Drift rate- occurs when the clocks do not run at an exact same speed. And this difference increases to a considerable level after some time and continues to be so.

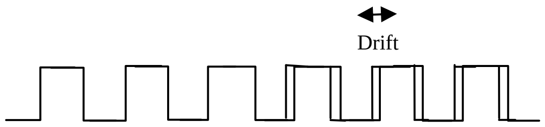


Fig. 4 Clock Drift

In the paper written by Kasim Sinan Yildirim [11], a tool is provided for finding the lower bounds of the distributed clock synchronization algorithms. And with the help of this tool the lower bound on the clock synchronization error between two processors in a distributed system can be proved.

In the method for clock synchronization proposed by Latha CA et al. the nodes in a distributed system are connected in the form of a ring. A Sync Token is a specific bit pattern that is made to rotate in the ring. Out of all the nodes in a ring one node is allowed to have a direct connection with the UTC server and this node is referred to as chief time server (CTS). In the beginning, CTS has the sync token and acts as a time server. It can then get the UTC value from UTC server. CTS then sets its clock value as the received UTC value and then broadcast that value to all the nodes connected in a ring [1].

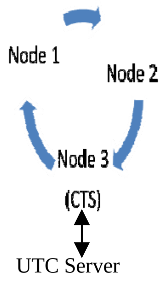


Fig. 5 Clock synchronization

4.2.1 Event ordering- for ordering of various events Lamport [7] defined a relation known as “happened before” and introduced the concept of logical clocks.

According to *happened before* relation on a set of events:

- If *a* and *b* are events in the same process, and *a* comes before *b*, then $a \rightarrow b$.
- If *a* is the sending of a message by one process and *b* is the receipt of the same message by another process, then $a \rightarrow b$.

- If $a \rightarrow b$ and $b \rightarrow c$, then $a \rightarrow c$.

4.2.2 Logical clock- lamport [7] gave the concept of logical clocks. A clock say C_i is associated with each process P_i and this process assigns a number $C_i(a)$ to any event a in that process.Logical clocks are a way to associate a timestamp to the events in a system. This allows the events to be properly ordered that are related to each other by the happened-before relation [2].

4.2.3 Vector clock- a vector clock system basically involves timestamping mechanism. Vector clocks were designed with the aim to allow processes to track the concurrency between the events produced by these processes [12]. The events that are produced by processes are mainly message sendings, message receives or internal events. A vector clock is defined as an array of n integers in which one entry is for one process for example if we take entry j it will tell us the number of events that are produced by the process p_j. Now the timestamp associated with each event defines the current value of the vector clock for a particular process that produced that event. With the help of these vector clocks it becomes easy to identify that whether two events are casually related or not.

Vector clocks are mostly preferred to achieve an extensible distributed framework like the Multi User Virtual Drawing Board because they provide global ordering and can solve various problems of casual broadcast, detection of message stability and detection of an event pattern [12]. A casual broadcast was first introduced by Birman and Joseph [12, 13].

The main problem of vector clock is scalability, R.Baldoni [12] in his paper presented a method to overcome this problem by giving the concept of Bounded Vector Clocks.

In the paper written by Li-Hsing Yen [14], a method for preserving the functionality of vector clock was given by performing correct clock resetting mechanism.

4.3 Deadlock- a deadlock is a situation where in the resources held by other processes are being requested by some other process in the same set [5].

Deadlock handling techniques:

4.3.1 Deadlock prevention- can be achieved by two ways:

→By allowing a process acquire all the resources it needs before it can actually begin execution. But this method involves various drawbacks like it effects the systems concurrency, may lead to certain processes enter into a deadlocked condition and also in certain systems it is difficult to actually predict what resources a process might need in future.

→By pre-empting the process that holds all the resources needed by a particular process. This method also have drawback like several processes are pre-empted without any deadlock.

4.3.2. Deadlock Avoidance- in case of distributed systems deadlock avoidance is achieved by granting resource to a process if the resulting system state is safe [5].This includes various drawbacks and makes it impractical to be used in distributed systems:

→In order to keep account for the safe state of the system by every site, large storage capacity and wide communication capability is required.

→A condition might occur where in several sites perform the safe state checking but the resulting net state may not be safe.

→This method for performing check for safe state is computationally expensive when large number of processes and resources are involved.

4.3.3. Deadlock Detection-Involves studying the relationship between the process and the resource to identify the presence of cyclic wait [5]. In distributed systems deadlock detection is advantageous in comparison to deadlock prevention and deadlock avoidance due to following reasons:

→Detecting a cycle in the system does not hinders the usual activities of a system.

→When a cycle is formed in the state graph it remains there in the system. Until it is detected and broken.

Issues in deadlock detection:

- State graph maintenance.
- To search a state graph for the presence of cycles.

Deadlock detection algorithms are divided into three parts:

4.3.3.1. Centralized deadlock detection algorithm- in centralized algorithms for deadlock detection, a control site is there that maintains the state graph for the entire system and performs the checks for the existence of any deadlock cycle. All sites are capable to request or release resources by sending "request resource message' and "release resource message". The control site updates its state graph whenever it receives such a message. This algorithm is though simple and feasible to implement but it is inefficient because it requires all the messages to go all way to the control site which in turn causes long delays for user request, traffic problem near the control site. All these reasons make this algorithm unreliable because everything depends on the control site [5].

Ho-Ramamoorthy presented two centralized deadlock detection algorithm :

i. Two-phase algorithm- a status table is maintained by every site in order to record the status of all the processes initiated at that particular site. A state graph is constructed by a designated site by requesting the state table from all the sites and it is then checked for the presence of cycle. If no cycle is formed then the system is not deadlocked. Then this designated site again constructs a state graph using only the transactions common to both reports by requesting status table from all sites. A system is declared deadlocked if the same cycle is detected again [5]. In this algorithm there may be a possibility of detecting a false deadlock so it is not that efficient.

ii. One-phase algorithm- involves only one phase of status reports from each site. However, two status tables are maintained by each site one for resource status and another for process status. The resource status table records all the transactions that have locked or are waiting for the resources stored at a particular site. The process status table keep a record of all the resources locked by or the resources that are being waited by all the transactions at a particular site [5]. A state graph is constructed by a designated site, by requesting both the tables from all the other sites and by using only the transactions that are same for both the resource table and process table and then performs check for the cycle. This algorithm does not detect the false deadlocks as it eliminates any inconsistency in state information by using only the information common to both tables. The one-phase algorithm is efficient in comparison to two-phase algorithm.

Table 1: Deadlock detection algorithms in distributed systems

Basis for distinction	Centralized Algorithm	Distributed Algorithm	Hierarchical Algorithm
1.State graph	Maintained at a single site called control site.	Distributed over several sites.	Sites are arranged in form of hierarchy.
2.Implementation	Easy	Difficult	Intermediate
3.Deadlock resolution	Simple	Cumbersome	Deadlocks are localized to as few clusters as possible.
4.Single point of failure	Yes	No	Employs to get best of both centralized and distributed.
5.Example	Ho-Ramamoorthy algorithm	Goldman's algorithm, Isloor-Marsland algorithm etc.	Menasce-Muntz algorithm.

4.3.3.2. Distributed deadlock detection algorithm- the work of detecting the deadlock cycle is distributed to all sites as all the sites co-operate each other to detect a deadlock cycle.

Various distributed deadlock detection algorithms are:

i. Goldman's algorithm- involves the use of an ordered blocked process list (OBPL), for exchanging deadlock related information. In OBPL each process is blocked by its successor and the last process in it will be either waiting to get an access to a particular resource or will be in a running state [5]. In this algorithm deadlock is detected by expanding the OBPL by attaching the process that holds the resource that is needed by the last process in the OBPL list at the end and this process is continued until a deadlock is detected or the OBPL is removed. Advantage of Goldman's algorithm is that whenever deadlock detection is to be carried out then only OBPL list is constructed.

ii. Isloor-Marsland algorithm- is also known as "online" deadlock detection algorithm. This algorithm detects deadlocks at the time of making decisions about resource allocation at a particular site. A reachable set for a particular node is constructed that consists of all the nodes that can be reached from it. A specific process is deadlocked if in a reachable set of a particular node, consists of that node also in the set. Deadlocks are detected by constructing reachable sets for all nodes. Each and every site maintains a state graph for the system and also the reachable sets for each node in the state graph.

iii. Obermarck's algorithm- provides a method to detect multisite deadlocks without the necessity to maintain a huge global transaction wait for graph (TWF) that is to be stored at each site. A node called "external" or Ex is a basic entity involved in this algorithm that abstracts nonlocal portion of the global TWF graph. Ron Obermarck [15] in his paper gave a proof of correctness for this algorithm of distributed deadlock detection including an example of the algorithm in operation along with the performance characteristics of the algorithm.

The process for deadlock detection is as follows:

$\longrightarrow$A particular site waits for deadlock-related information to be received from other sites.

$\longrightarrow$As soon as the information is received by this site it then combines this information with its local TWF graph. And then detects all cycles and the cycle that does not contain the node Ex is broken.

$\longrightarrow$ If the node Ex is contained in the cycles $Ex \longrightarrow T_1 \longrightarrow T_2 \longrightarrow Ex$, the particular site sends them in a string form Ex, T_1, T_2, Ex to all the other sites.

In this algorithm message traffic is reduced and a string say Ex, T_1, T_2, T_3, Ex is sent to other sites only if T_1 is higher than T_3 in the lexical ordering [5].

iv. Chandy- Misra- Haas algorithm- a concept of special message called probe is used in this algorithm. A probe is defined by a set (i, j, k). It means that in this set

a deadlock detection is initiated for process P_i that is being sent by home site of process P_j to the home site of process P_k. The probe message travels along the edges of the global TWF graph and we say deadlock is detected when this message returns to its initiating process.
Chandy, Misra and Haas [16] in their paper presented a distributed deadlock models and also said that no false deadlocks are reported.

4.3.3.3. Hierarchical deadlock detection algorithm- in this algorithm all the sites are arranged in hierarchical manner. A particular site is responsible for detecting deadlocks that involves only its children sites [5].

i. Menasce-Muntz algorithm- in this algorithm all the resource controllers are arranged in the form of a tree. In the hierarchy the controllers that manage resources are locate at the lowest level, and are referred to as leaf controllers. These leaf controllers also perform the task of deadlock detection and also maintains the part of the global TWF graph that is concerned with resource allocation at the leaf controller. A TWF graph is maintained by a non leaf controller that spans only its children controller and detects deadlocks that involves its own leaf controllers. Any change in the TWF graph of a controller as a result of resource allocation, wait, or release causes an appropriate change in its parent controller [5]. Now this parent controller makes appropriate changes in the TWF graph and searches for any cycle. And if necessary these changes are sent up in the hierarchy.

ii. Ho-Ramamoorthy algorithm- in Ho and Ramamoorthy 's hierarchical algorithm the sites are grouped to form several clusters. A site is periodically chosen as the central control site and it then dynamically chooses control site for each cluster. This central site can request other control sites for their inter cluster transaction status information and wait-for relations. With the help of this process at a particular control site, status tables of all the sites is collected in its cluster and then one-phase deadlock detection algorithm is applied to detect all deadlocks involving only intra cluster transactions. This site then sends the inter cluster transactions status information and wait-for relations to the central site. This helps the central site to construct the system state graph, and thus perform checks for the presence of any cycle. Therefore, it can be summed up and said that the control site detects deadlocks that are present in its own cluster and central site detects all inter cluster deadlocks.
Edgar Knapp [17], in his paper presented basic principles on which distributed deadlock detection schemes are based. He also said that these principles provide a method to develop distributed algorithms. In his paper he discussed number of algorithms and also their respective complexities.

Out of all the deadlock detection algorithms discussed so far, for Multi User Virtual Drawing Board the distributed approach is the best to be used in order to achieve an extensible distributed framework. All the distributed deadlock algorithm aims to detect cycles not only at a particular site but spans several sites in the system. Now the way this aim is achieved by distributed deadlock detection algorithm differs according to the method used that is the type of distributed algorithm used (Goldman's, Obermarck's, Isloor Marsland's etc.)
Therefore, it depends on the type of distributed deadlock detection algorithm we are using that have its associated advantages and disadvantages that makes an application more reliable and secure.

4.4 Concurrency management- several processes in a distributed system are said to be concurrent if they perform their task at the same time. Now this concurrency may give rise to several problems like 'inconsistent update' problem and 'inconsistent retrievals' problem [6]. Inconsistent retrieval refers to the situation when a particular transaction reads some data objects of a database before another transaction has completed some modifications on those data objects [9]. Inconsistent update refers to the situation when on a common set of data objects many transactions perform read and write that leads to the inconsistency in the database [9].
Various concurrency control algorithms are:

4.4.1. Locking- is most widely used algorithm for concurrency control. In locking method a transaction locks a data item before actually accessing it. Now this transaction can access this data item any number of times. And no other transaction can have access to this data unless it is released by the transaction that acquired a lock over it[2].

4.4.2. Optimistic concurrency control- was given by Kung and Robinson [8]. This method allows the transactions to continue until the end of first phase. But in the second phase the before a transaction is committed , the transaction is first validated to check any inconsistency caused by any other transaction since it is started. The transaction is committed only if it is found valid otherwise it is aborted.

4.4.3. Timestamps- in this method a transaction is assigned a unique timestamp at the time it is started. Every data is assigned two timestamps that is *read timestamp and write timestamp.* Whenever a transaction wants to access a data then the data item's read timestamp or write timestamp is updated according to the transactions timestamp depending on the type of access [2].
Ricart and Agrawala in 1981 [18] said that if a process wants to enter the critical section it should ask the other processes to give it the permission to enter into the

critical section. Therefore, it waits for the appropriate permissions for the same. For this reason Ricart-Agrawala proposed the timestamp mechanism in order to associate a timestamp for each request. Since these timestamps are ordered therefore, conflict problem is resolved. The request having the lowest timestamp amongst all the conflicting requests gets the highest priority and by this conflicts are resolved.

Michel Raynal [19], in his paper discussed the principles from which the distributed mutual exclusion algorithms are designed. These principles include permission-based and token-based principles.

Permission-based algorithms include the one that was given by Ricart and Agrawala that is based on timestamps.

In token-based method, tokens are used to grant permissions to the processes that want to enter into the critical section and these processes waits until the token arrives.

Kerry Raymond [20] in his paper proposed a tree-based algorithm for distributed mutual exclusion. According to this method in a system there are *N* nodes that communicate to each other by means of passing messages. In this algorithm a spanning tree of the system is used. The number of messages exchanged per critical section depends on the topology of the tree. Each node in this tree has the capability of storing information about their immediate neighbour rather than about all nodes that are not in the spanning tree. The nodes that fail can recover all the necessary information from their neighbours. This algorithm does not require the use of sequence numbers because it operates correctly without the use of it also.

Suzuki and Kasami [21] in their paper gave a distributed mutual exclusion algorithm. In this algorithm mutual exclusion is carried out among *N* nodes in a system and it requires N message exchanges for each mutual exclusion invocation. With this algorithm the delay that was caused to invoke mutual exclusion is much smaller in comparison to the algorithm for mutual exclusion given by Ricart and Agrawala in which 2*(*N*-1) message exchanges are required per invocation. The drawback of this algorithm given by Suzuki and Kasami is that it uses sequence number concept and these sequence numbers contained in the messages are unbounded but in this paper a method is given to resolve this problem by slightly increasing the number of message exchanges.

For Multi User Virtual Drawing Board concurrency management can be done by applying serialization protocol in order to perform simultaneous updates.

4.5 Consensus- a consensus problem is a situation where each and every processor broadcasts its initial value to all the other processors in the system [9]. Now this initial value may be different for all the processors.

So a protocol for reaching consensus is required and this protocol should meet the following conditions:

→*Agreement:* all the processors that are non-faulty must agree on the same single value.

→*Validity:* now if the non-faulty processors have an initial value of *v* then the common value on which the processors must agree should be *v*.

Michael J. Fischer *et.al* [10] in his paper discussed about consensus problem. In his paper he said that consensus problem involves an asynchronous system of processes and some of these processes may be unreliable. He also discussed about "Byzantine Generals" problem.

For Multi User Virtual Drawing Board consensus problem can be solved by using a primary replication server which can be used to continuously send updates and check for acknowledgement.

Table 2: Distinction between three Agreement problems in Distributed Systems

	Byzantine Agreement	**Consensus**	**Interactive Consistency**
1.Initiator of the value	One processor initiates the value	All processors	All processors
2.Final agreement	Single value	Single value	A vector of values
3.solution of the problem	Solved by solution to the consensus problem	Solved by solution to the interactive consistency problem.	Byzantine agreement problem is a primitive to solve this problem.

4.6 Security in distributed systems- since in a distributed system, nodes are located at various locations. So a major concern in a distributed system is the security of an application. Robert Cole [22] in his paper gave a model of security in distributed systems. And also certain issues on the use of this security model were given.

There are various threats that are needed to be addressed in a distributed system [22] like information disclosure, use of resources by unauthorised means, repudiation of information flow, denial of service, information contamination, misuse of resources.

A model of security in a distributed system should be such that it should cover all the security threats and this model should well adapt to suit different security policies. He also defined the concept of security information that is always generated by a specific security services in response of an authenticated requests. And the main purpose of any security service is to use security information appropriately in a distributed system [22].

Dan M. Nessett [23] in his paper gave certain factors that affect the security of distributed systems including factor like node evaluation levels and network topology.

Rob Dobry [24] in his paper discussed various aspects of security required in distributed systems. The various aspects include cryptography, data confidentiality, and data integrity. He said in order to develop a secure distributed system , a system should not only make use of traditional computer security concepts but must also utilize communication security concepts.

Alan H. Karp [25] in his paper gave three components of the system architecture to make it easier to manage and monitor distributed systems. The work he presented was based on certain assumptions like large number of machines and users, dynamic, heterogeneous, hostile, different environments. The three components used in this paper were separate granting of rights from access control, mediate between applications and user and lastly using a proxy for remote users.

Security in MUVDB will be achieved in a way that no user should be able to delay network traffic or cause denial of service and during consensus a user is not able to vote multiple times. We handle the first issue by diagnosing abrupt traffic generating nodes and further eliminating them from our multicast network. The second issue is handled by authenticating users with user database and allowing only authentic users to participate in the application. We also make voting during consensus as an idempotent operation.

5. Conclusion

A distributed system consists of various computers located at different sites. Each of these computers have their own local memory and processors. All these computers are connected by means of a communication network. As seen through this paper that there are several issues related to the distributed system and all these issues have their respective advantages and disadvantages. A message passing system should be such that it should deliver the message from sender to receiver without leading to any type of error. Clocks must be synchronized to avoid the problem of clock skew and clock drift. And for a distributed system it is appropriate to use vector clocks for global ordering of events. As already discussed that amongst the three deadlock handling techniques deadlock detection is most advantageous in comparison to deadlock avoidance and deadlock prevention and there are various deadlock detection algorithms that can be applied depending upon the need of the system. In case of concurrency management through this paper it can be concluded that timestamps technique for concurrency control is better in comparison to the other two techniques discussed in the paper. As far as consensus problem is concerned it is clear through the table for the three agreement problems that all these agreement problems are complementary to each other. The security of the distributed system is the key aspect while developing any distributed system. Security includes various threats that should be addressed whenever distributed systems are taken into account.

6. Future Scope

The idea presented in this paper for an extensible distributed framework can be implemented to develop a Multi User Virtual Drawing Board. A Multi User Virtual Drawing Board can support multiple users at a same time to solve the problem of collaborative designing. The idea behind this Multi User Virtual Drawing Board is to provide users with a virtual drawing board and all changes made by each user are reflected in all the virtual boards. For this various concepts of distributed system can be used like message passing, sending updates, making consensus for agreement in case of conflicts, concurrency management etc. Interface for such an application can be designed using swings and a message passing model using sockets and threads.

References

[1] Latha CA, Dr. Shashidhara HL, "Clock Synchronization in Distributed Systems", *2010 5th International Conference on Industrial and Information Systems, ICIIS* 2010, Jul 29-Aug 01, India.

[2] Pradeep K. Sinha "Distributed Systems" concepts and design, *Prentice-Hall, India.*

[3] Kenneth P. Birman, "The Process Group Approach to Reliable Distributed Computing", *communication of the ACM* December 1993/vol.36, no.12.

[4] Hagit Attiya, Amotz Bar-Noy and Danny Dolev, "Sharing Memory Robustly in Message-Passing Systems", *journal of the association for computing machinery,* vol 42, no. 1, January 1995, pp 124-142.

[5] Mukesh Singhal, "Deadlock Detection in Distributed System", *Ohio State University, November 1989, IEEE.*

[6] George Coulouris, "Distributed Systems concepts and design", *fourth edition, pearson education.*

[7] Leslie Lamport, "Time, Clocks, and the Ordering of Events in a Distributed System, *communication of the ACM,* july 1978,vol 21,no. 7.

[8] Kung and Robinson, "On Optimistic Methods for *Concurrency* Control", *ACM transactions on database systems,* vol. 6, no. 2, pp. 213-226 (1981).

[9] Mukesh Singhal, Niranjan G. Shivaratri, "Advanced Concepts in Operating Systems", *Tata McGraw-Hill edition.*

[10] Michael J. Fischer, Nancy A. Lynch and Michael S. Paterson, "Impossibility of Distributed Consensus with One Faulty Process", *journal of the Association for Computing Machinery, vol. 32, no. 2.*
[11] Kasim Sinan Yildirim and Aylin Kantarci, "Clock Synchronization in Distributed Systems", Computer Engineering Department, Ege University. *The Turkish Scientific and Technical Research Council (TUBITAK).*
[12] R.Baldoni, M.Raynal, "Fundamentals of Distributed Computing: A Practical Tour of Vector Clock Systems".
[13] Birman K and Joseph T, "Reliable Communication in the presence of failures". ACM Transactions on Computer Systems, 5(1): 47-76, 1987.
[14] Li-Hsing Yen and Ting-Lu Huang, "Resetting Vector Clocks in Distributed Systems", *Journal of Parallel and Distributed Computing 43, 15-20 (1997, Article No.PC971330.*
[15] Ron Obermarck, "Distributed Deadlock Detection Algorithm", *ACM Transactions on Database Systems, vol.7, No.2, June 1982, 187-208.*
[16] K. Mani Chandy, Jayadev Misra and Laura M. Haas, "Distributed Deadlock Detection", *ACM Transactions on Computer Systems, Vol. 1, No. 2, May 1983, 144-156*.
[17] Edgar Knapp, "Deadlock Detection in Distributed Databases", *ACM Computing Surveys, Vol. 19, No. 4, December 1987.*
[18] Ricart G. Agrawala A.K, "An Optimal Algorithm for Mutual Exclusion in Computer Networks", *Comm.ACM, Vol. 24, 1, (1981) , pp.9-17.*
[19] Michel Raynal, "A Simple Taxonomy for Distributed Mutual Exclusion Algorithms".
[20] Kerry Raymond, "A Tree-Based Algorithm for Distributed Mutual Exclusion", *ACM Transactions on Computer Systems, Vol. 7, No. 1, February 1989,pages 61-77.*
[21] Ichiro Suzuki and Tadao Kasami, "A Distributed Mutual Exclusion Algorithm", *ACM Transaction on Computer Systems, Vol. 3, No.4, November 1985, pages 344-349.*
[22] Robert Cole, "Security for Distributed Systems", *Hewlett-Packard Laboratories.*
[23] Dan .M. Nessett, "Factors Affecting Distributed System Security", *IEEE Transactions on Software Engineeering, Vol. SE-13, No. 2, February 1987.*
[24] Rob Dobry and Mary D. Schanken, "Security Concerns for Distributed Systems", *IEEE 1994.*
[25] Alan H. Karp and Kevin Smathers, "Three Design Patterns for Secure Distributed Systems", *Hewlett-Packard Company 2003.*

Aneesha Sharma is a student who has received her B.Tech degree in Computer Science and Engineering in 2010 with first division from Amity School of Engineering and Technology, Amity University, Noida, Uttar Pradesh, India. Currently, the author is pursuing M.Tech in Computer Science and Engineering from Amity School of Engineering and Technology, Amity University, Noida, Uttar Pradesh, India. The author is currently undergoing a dissertation period in the field of Distributed Systems.

Shilpi Gupta is working as an assistant professor in Department of Computer Science and Engineering of Amity School of Engineering & Technology, Amity University, Noida. She has 05 years of experience in the field of Academics and is actively involved in research & development activities. She has received her B.Tech degree in 2006. She is M.Tech Gold medalist in the stream of Computer Science and Engineering from Jaypee Institute of Information Technology, Noida. Her area of interest includes Software Engineering, Artificial Intelligence, Soft Computing, Cognitive Informatics, Affective Computing. She has successfully published national and international research papers.