PERFORMANCE ANALYSIS OF QUORUM BASED ALGORITHMS IN CELLULAR NETWORKS

A Dissertation submitted in partial fulfilment of the requirements
for the award of the degree of

MASTER OF ENGINEERING
In
(Computer Technology & Applications)

Submitted By:

RACHITA JUNEJA

(University Roll No. 8550)

(College Roll No. 11/CTA/09)

Under the Guidance of

Mr. Vinod Kumar

Department Of Computer Engineering

Delhi College Of Engineering, Delhi

DEPARTMENT OF COMPUTER ENGINEERING

DELHI COLLEGE OF ENGINEERING

DELHI UNIVERSITY

2011

CERTIFICATE

DELHI COLLEGE OF ENGINEERING
DELHI UNIVERSITY
2010-2011

This is to certify that the work contained in this dissertation entitled "Performance Analysis of Quorum Based Algorithms in Cellular Networks" submitted by Rachita Juneja, University Roll No. 8550 in the requirement for the partial fulfillment for the major project in Master of Engineering in Computer Technology & Applications, Delhi College of Engineering is an account of her work carried out under my direct supervision and guidance in the academic year 2010-2011. She has completed her work with utmost sincerity and diligence.

(Mr. Vinod Kumar)
Assistant Professor
Project Guide
Dept. of Computer Engineering
Delhi College of Engineering
Delhi

ACKNOWLEDGEMENT

It gives me a great pleasure to express my profound gratitude to my project guide Mr. Vinod Kumar, Faculty, Department of Computer Engineering, Delhi College of Engineering, for his valuable and inspiring guidance throughout the progress of this project. He kept on boosting me time to time for putting an extra ounce of effort to realize this work.

At the same time, I would like to extend my heartfelt thanks to Dr. (Mrs.) Daya Gupta, Head of the department, Department of Computer Engineering, Delhi College of Engineering, for keeping the spirits high and clearing the visions to work on the project.

I would also like to take this opportunity to present my sincere regards to all the teachers in the Department of Computer Engineering for their support and encouragement.

I would also like to thank my classmates for their unconditional support and motivation during this work.

Lastly, I would like to thank Almighty GOD for his countless blessings.

RACHITA JUNEJA

Master in Engineering

(Computer Technology & Applications)

University Roll No. 8550

Dept. Of Computer Engineering

Delhi College of Engineering

CONTENTS

List of Figures

ABSTRACT

The assigned frequency spectrum to the wireless mobile systems has become a scarce resource as the number of mobile users has increased tremendously. So there is a need of using the allotted bandwidth efficiently. Distributed Mutual Exclusion assigns channels to various cells and increases bandwidth utilization. Various algorithms exist which help in attaining mutual exclusion. Quorum based algorithms is one such class of algorithms where the requesting site asks permission from a set of smaller number of participating sites called a quorum. Quorums help reduce the message complexity in mobile systems. Performance analysis of quorum based algorithms is done using message complexity, synchronization delay, response time system throughput as the metrics.

Chapter 1: Introduction

1.1 Distributed systems

A distributed system is a collection of autonomous computers connected via a communication network. There is no common memory and processes communicate through message passing. One of the most important purposes of the distributed systems is to provide an efficient and convenient environment for sharing of resources. They also provide computational speedup and better reliability. In a system consisting of a number of processes, each process has a segment of code, called a critical section, in which the process may be changing common variables, updating a table, writing a file and so on. A distributed system may have thousands of data items and scores of databases residing in many widely dispersed sites. Moreover, many users can have simultaneous access to this data. Therefore, it is required that processes at different sites cooperate with one another and have some sort of coordination among them. In many applications, there are critical operations that should not be performed by different processes concurrently. It may sometimes be required that a typical resource is used by only one process at a time. Or, if a data item is replicated at various sites, we expect that only one of the sites updates it at a time. ***This gives rise to the problem of mutual exclusion, which requires that only one of the contending processes be allowed, at a time, to enter its critical section (CS).*** Thus, mutual exclusion is crucial for the design of distributed systems.

An important application of distributed systems, which uses mutual exclusion, is in the field of replicated databases. Replication is the maintenance of on-line copies of data and other resources. A replicated database is a distributed database in which some data items are stored redundantly at multiple sites. In replicated databases, a data item or a file is replicated at a number of sites with independent failure modes. This results in increased availability, better fault-tolerance capability and improved response time. However, maintaining the consistency of the data is a major issue. For example, owing to node or link failures, the network may partition into isolated groups of nodes. If a node is to perform updates, it must ensure that no other node is doing so. Thus, mutual exclusion is needed for updating files in replicated databases. Protocols that control access to replicated data and ensure data consistency in case of network partitioning are called replica control protocols. A transaction which is an access request to data is the basic unit of user computation in a database. It executes in three steps: it reads a portion of the database into a local workspace; then it performs some local computation on it; and finally, it writes some values into the database. All replica control protocols require that mutual exclusion must be guaranteed between two write operations and a read and write operation.

Other areas where mutual exclusion is desirable include atomic commitment, cellular networks. A general idea about geographical division of cellular communication network consists of clusters of hexagonal cells each having a fixed base station called mobile service station (MSS). All the MSS's are connected to each other through a fixed communication network. When a mobile host (MH) wants to establish a call, it sends a request to MSS. If a free channel is available with the MSS, it grants the channel to the MH which then proceeds with the call. If a

particular channel is being used by more than one MH at the same time in a cell or the neighbouring cells, the calls will interfere with each other. Such interference is called co-channel interference. The use of particular frequency in a cell for communication session establishment can be viewed as equivalent to entering the critical section by the cell in which the channel is being used. In mobile communication, frequency channels are the common resource. Each cell has to attain a frequency channel for communication. Since two or more neighbouring cells can try to attain the same channel at the same time, this can be viewed as similar to Mutual Exclusion Problem where the processes wait for a shared resource currently being used by some other process in order to complete their task.

The bandwidth assignment in a mobile cellular system can be static, dynamic or the hybrid of the two. Distributed Dynamic Channel Allocation can be considered as an application of ME. Here a cell wants to be assigned a channel for call establishment. It does not matter from where the channel is obtained as long as there is no interference between various calls.

In distributed ME, the critical resources need to be assigned to different sites such that at a particular site, two or more processes cannot use the same critical resource at the same time. Also such a resource can be used among various different sites at the same time as long as they are non- interfering. So distributed ME assigns channels to various cells and increases bandwidth utilization and at the same time reduces channel interference.

1.2 History of mutual exclusion

The origin of the mutual exclusion problem can be traced back to 1965 when Dijkstra described and solved the mutual exclusion problem. Dijkstra stated that any

solution to the mutual exclusion problem must satisfy 4 constraints. Dijkstra's algorithm guaranteed mutual exclusion and was deadlock- free, but it did not guarantee fairness. That is, it was possible that a process seeking access to its critical section waited indefinitely while others entered and exited their critical sections frequently. Knuth proposed the first fair solution to the mutual exclusion problem. In addition to the four constraints specified by Dijkstra, Knuth's algorithm also satisfied the following:

(i) No assumption is made about the instructions or the numbers of processors supported by the machine except that basic instructions such as ead, write or test a memory location are executed atomically.

(ii) No assumption is made about the execution speed of the competing processes, except that it is nonzero.

(iii) When a process is in the noncritical section, it cannot prevent another from entering the critical section.

(iv) It must not be possible for a process seeking access to a critical section to be delayed indefinitely.

(v) A process attempting to enter its critical section will be able to do so within a finite time.

Thereafter, a number of algorithms were proposed which guaranteed mutual exclusion and were deadlock- free and fair. Each of these algorithms aimed at improving performance in terms of synchronization delay, the period of time between the instant a site invokes mutual exclusion and the instant when it enters CS. Some of these algorithms are De Bruijns's algorithm, Eisenberg and McGuire's algorithm and Peterson's algorithm. All these algorithms were designed for the centralized systems, the systems possessing a central memory that all processes can access

simultaneously for reading and writing. Their basic principle was the same: they all used one or more shared variables to achieve mutual exclusion, just similar to the case of single processor systems, where critical regions are protected using semaphores, monitors etc.

But in the distributed systems due to lack of a common shared memory, the problem of mutual exclusion becomes much more complex as compared to the centralized systems and needs special treatment. The algorithms designed to ensure mutual exclusion in distributed systems are termed distributed mutual exclusion (DME) algorithms. A number of distributed mutual exclusion algorithms have been proposed. They have been classified as token-based algorithms or permission-based algorithms depending on the technique used to achieve mutual exclusion. A good DME algorithm, besides providing mutual exclusion, should take care that there are no deadlocks and starvation does not occur.

1.3 Organisation of Thesis

The rest of the thesis is organized as follows:

Chapter 2 deals with the classification of mutual exclusion algorithms and details of distributed ME.

Chapter 3 deals with the definition and properties of quorums along with various quorum construction algorithms.

Chapter 4 deals with details of a hexagonal cellular network, reuse distance, co channel interference and various channel assignment strategies.

Chapter 5 is about the work done under the thesis.

Chapter 6 is the conclusion of the thesis and future work.

Finally, details of references made have been listed.

Chapter 2: Mutual Exclusion

2.1 Definition

When a process executes the shared variable, all other processes desiring to do so at the same time are kept waiting till that process has finished executing the shared variable. This is called Mutual Exclusion. Mutual exclusion means that the processes execute the critical section in time intervals that do not overlap. Any facility which has to provide support for mutual exclusion should meet the following requirements:

1. Only one process at a time is allowed into its critical section, among the processes that have critical sections for the same resource or shared object. To impose some form of control upon the execution of critical sections, the following programming structure is used:

enter_critical();

/* critical section */

exit_critical();

where enter critical() and exit critical() are provided by the system in some way and they together guarantee the exclusive access to shared resources.

2. When no process is in a critical section, any process that requests entry to its critical section must be permitted to enter without delay.

3. A process remains inside its critical section for a finite time only.

4. No assumptions should be made about relative process speeds or number of processors. That is the facility provided should be powerful enough to solve

the control problems unconditionally, without any limit on the nature of concurrent processes.

2.2 Classification of Mutual Exclusion Algorithms

Algorithms for mutual exclusion may be Centralized or Distributed.

In a centralized system, one node is designated as the control node. This node controls has access to all shared objects. Whenever a process wishes to gain access to the shared object, it sends a request message to the control process, which returns a reply message when the shared object becomes available. Only the control node makes resource-allocation decisions. It uses Request, Permission, and Release messages. Mutual exclusion can be achieved in this scheme. The coordinator (control node) lets only one process at a time into each CS. It is fair as requests are granted in the order in which they are received. No process waits forever. The centralized algorithm is simplest and also most efficient. It requires only three messages to enter and leave a critical region: a request, a grant to enter, and a release to exit. A centralized mutual exclusion algorithm is characterized by the following properties:

- Only the central (control) process makes a decision.

- All necessary information is concentrated in the central (control) process.

But the control node may be a bottleneck. Failure of the control node causes a breakdown of mutual exclusion.

The problem of mutual exclusion becomes much more complex in distributed systems (as compared to single-computer systems) due to the lack of both a shared memory and a common physical clock and because of unpredictable message

delays. . In a distributed system, shared variables or a local kernel cannot be used to implement mutual exclusion. Message passing is the sole means for implementing distributed mutual exclusion. The design of a mutual exclusion algorithm consists of defining the protocols used to coordinate access to a shared object. Each node has only a partial picture of the total system and must make decisions based on this information. A distributed algorithm for mutual exclusion is characterized by these properties:

- All processes have an equal amount of information.

- All processes make a decision based on local information.

- All nodes bear equal responsibility for the final decision.

- There is no common clock and no way to adequately synchronize clocks

- Failure of a node, in general, does not result in a total system collapse.

Distributed algorithms can be subdivided into:

–**Token based**: A site is allowed in the CS if it has a token. Tokens are passed from site to site, in some priority order.

– **Non-token based**: Two or more successive rounds of messages are exchanged among the sites to determine which site will enter the CS next.

–Quorum based: Each site requests permission to execute the CS from a subset of sites (called a quorum). Any two quorums contain a common site. This common site is responsible to make sure that only one request executes the CS at any time.

Non token based and Quorum based algorithms belong to a class of DME algorithms called the **permission based** algorithms.

2.3 Distributed algorithms

Distributed algorithms require a time ordering of events. An “event” is the sending of a message. In such systems mutual exclusion can be achieved without deadlock or starvation. All processes are involved in all decisions. But the single point of failure has been replaced by *n* points of failure. If any process crashes, it will fail to respond to requests. This silence will be interpreted incorrectly as denial of permission..Each process must maintain the group membership list itself, including processes entering the group, leaving the group and crashing.

Many distributed algorithms for mutual exclusion have been proposed. In Lamport's algorithm each process has a queue. A process which wants to enter critical section, broadcasts a request message with a time-stamp. The return of time-stamped acknowledgements allows it to check whether a process has invoked the critical section earlier than itself. The number of exchanged messages is 3(N-1), where N is the number of processes in the network. Lamport’s time stamping algorithm gives a consistent time-ordering of events in a distributed system. Each node I has a local counter C_I. When node I sends a message, it first increments C_I by 1. Messages from node I all have form (m, T_I, I), where m is the actual message (like Request or Release), I is the node number, T_I is a copy of C_I (the node’s “timestamp”) at the time the message was created. When node J receives a message from node I, it updates its local C_J to 1 + max { C_J , T_i }. (m, T_I, I) precedes (m, T_J, J) if $T_I < T_J$.

if $T_I = T_J$ and I < J

For this to work, each message must be sent to all other nodes.

In Ricart & Agrawala algorithm, if a process wants to enter critical section, it composes a message containing:

- Identifier (machine ID, process ID)
- Name of resource
- Timestamp (totally-ordered Lamport)

It sends request to all processes in group and waits until everyone gives permission. Then the process enters the critical section. When a process receives request, it sends OK to sender if the receiver is not interested. If the receiver is in critical section, it does not reply and add the request to its queue. If the receiver also just sent a request, then it compares the timestamps of the received & the sent messages. The earliest among them wins. If the receiver is the loser, it sends **OK** to the sender. If receiver is the winner, it does not reply and queues the senders request. When the receiver is done with critical section, it sends OK to all queued requests**.**

Ricart and Agrawala algorithm demonstrates that a fully distributed algorithm is possible but it has N points of failure and a lot of messaging traffic between the nodes.

Difference between Lamport and Ricart-Agrawala algorithm:

- Every node in the system responds always in Lamport's algorithm and there is no hold-back if a receiver node cannot give permission to the sender.
- Node decides to go based on whether its request is the earliest in its queue.

- The distributed algorithm by Ricart and Agrawala requires (n – 1) request messages, one to each of the other processes, and an additional (n – 1) grant messages, for a total of 2(n − 1) while the Lamport's algorithm requires 3(n - 1) messages.

CHAPTER 3: QUORUM BASED ALGORITHMS

There are two classes of distributed mutual exclusion algorithms namely Token based and Permission based mutual exclusion algorithms.

Token based algorithms achieve ME using a privilege message called token which is shared among all the participating sites. Since there is a single token for the entire system ME is guaranteed. No two sites can possess the token at the same time. The token based algorithms do not find effective use in channel allocation as frequency reuse is being done in order to maximise bandwidth utilization. A single token should exist in a system is the basis of token based algorithms which is violated in channel allocation (as frequency reuse is applicable here).

Permission based algorithms need permission from participating sites in order to execute critical section. When a process in a site wants to enter CS, it has to acquire permission from all the participating sites. If all these sites agree, only then can a process enter CS. When some of the participating sites themselves want to enter CS, timestamps are used to avoid conflict. If the timestamp of requesting site is higher than any of the participating sites which also want to execute CS, it is not granted permission.

The advantage of permission based mutual exclusion algorithms is that they exhibit excellent fault-tolerance and load-balancing characteristics. The main drawback of permission based mutual exclusion algorithms is that the communication cost to enter critical section is directly proportional to the size of quorums.

3.1 Classification of permission based algorithms

Permission based algorithms can further classified as voting based and coterie based algorithms. In voting based algorithms each participating site is assigned a non- negative integer called a vote. Permission needs to be taken from the participating sites such that the sum total of votes acquired by a requesting site is simple majority to the total number of votes in the system. Thus the voting algorithms use majority voting to achieve ME. A site asking for permission to execute CS does not worry as to which sites vote for it. What it really worries is that it should get majority votes in the system. Such a scheme of majority voting given by Thomas assigns uniform votes (i.e. equal weight age) to the entire system. Another scheme given by Gifford uses weighted voting in which some of the participating sites in the system are assigned different votes than other sites. This technique is called quorum consensus method.

In coterie based permission algorithms, a sub-group of sites is constituted according to some rule and the requesting site needs to acquire grant messages from this sub group only rather than the entire system. Such a sub- group is called a quorum. A quorum system (or a coterie) is a collection of these sub-groups (quorums), every two of these quorums intersect. For example, {{1, 3,5}, {2,3,6}, {4,5,6}, {1,2,4}} is a set of four quorums constituted from a universal set U={1,2,3,4,5,6}. Thus a coterie C is defined as a set of sets, where each set g ∈ C is called a quorum.

3.2 Properties of Coteries

The following properties holds for quorums in a coterie are explained in detail:

Intersection property: For every quorum g, h ∈ C, g ∩ h≠ ϕ. For example, sets {1, 2, 3}, {2, 5, 7} and {5, 7, 9} cannot be quorums in a coterie because the first and third sets do not have a common element.

Minimality property: There should be no quorums g, h in coterie C such that g ⊇ h. For example, sets {1, 2, 3} and {1, 3} cannot be quorums in a coterie because the first set is a superset of the second. This property suggests that no quorum is a superset of another quorum in the coterie. In another example, {{1, 3}, {1, 4}, {2, 3}, {2, 4}} is a set of four quorums formed from a universal set U={1,2,3,4} and is called a 2-coterie. In this coterie, {{1, 3}, {2, 4} or {1, 4}, {2, 3}} are two mutually disjoint quorums and minimality property is satisfied. This property helps in multiple entries to CS for mutually disjoint quorums.

3.3 About Quorums

Quorums decrease the message complexity in the system which is limited to the size of the quorum. Quorum systems have been used for many applications in the area of distributed systems, including mutual exclusion, data replication etc. Quorum-based mutual exclusion algorithms are different in the following ways:

1. A site does not request permission from all other sites, but only from a subset of the sites.
2. The request set of sites are chosen such that ∀i ∀j: 1 ≤ i, j ≤ N:

Ri ∩ Rj ≠ ϕ. Consequently, every pair of sites has a site which mediates conflicts between that pair.

Coteries and quorums can be used to develop algorithms to ensure mutual exclusion in a distributed environment. A simple protocol works as follows:

Let ‘a’ is a site in quorum ‘A’. If ‘a’ wants to invoke mutual exclusion, it requests permission from all sites in its quorum ‘A’. Every site does the same to invoke mutual exclusion. Due to the Intersection property, quorum ‘A’ contains at least one site that is common to the quorum of every other site. These common sites send permission to only one site at any time. Thus, mutual exclusion is guaranteed. The Minimality property ensures efficiency rather than correctness.

A mutual exclusion algorithm works as described here .Site s sends a ‘Request’ message to enter the critical section (CS) to all other sites in the structured quorum it belongs to. Each site in the quorum stores incoming requests in a request queue, ordered by their timestamps. A site sends a ‘Reply’ message, indicating its consent to enter CS, only to the request at the head of its request queue, having the lowest timestamp. If the site s gets a ‘Reply’ message from all sites in the structured quorum it belongs to, it enters the CS. After exiting the CS, s sends a ‘Release’ message to all sites in the structured quorum. On the receipt of the ‘Release’ message, each site removes s’s request from the head of its request queue. If a new request arrives with a timestamp smaller than the request at the head of the queue, an ‘Inquire’ message is sent to the process whose request is at the head of the queue and waits for a ‘Yield’ or ‘Release’ message. When a site s receives an ‘Inquire’ message, it acts as follows:

If s has acquired all of its necessary replies to access the CS, then it simply ignores the ‘Inquire’ message and proceeds normally and sends a ‘Release’

message after exiting the CS. If s has not yet collected enough replies from its quorum, then it sends a 'Yield' message to the inquiring site. When a site gets the 'Yield' message, it puts the pending request (on behalf of which the 'Inquire' message was sent) at the head of the queue and sends a 'Reply' message to the requestor. Mutual exclusion is guaranteed because the set of quorums satisfy the Intersection property.

3.4 Example:

Consider a coterie C which consists of quorums {1,2,3}, {2,4,5} and {4,1,6}. Suppose nodes 3, 5 and 6 want to enter CS, and they send requests to sites (1, 2), (2, 4) and (1, 4), respectively. Suppose site 3's request arrives at site 2 before site 5's request. In this case, site 2 will grant permission to site 3's request and reject site 5's request. Similarly, suppose site 3's request arrives at site 1 before site 6's request. So site 1 will grant permission to site 3's request and reject site 6's request. Since sites 5 and 6 did not get consent from all sites in their quorums, they do not enter the CS. Since site 3 alone gets consent from all sites in its quorum, it enters the CS and mutual exclusion is achieved.

3.5 Algorithms for quorum construction

Different algorithms exist for constructing quorums. Maekawa's algorithm was the first quorum-based mutual exclusion algorithm. The request sets for sites (i.e., quorums) in Maekawa's algorithm are constructed to satisfy the following conditions:

1. Q_i is contained in $S_i \forall\ i \in 1, 2, 3... N$
2. $S_i \cup S_j \neq \emptyset \forall\ i, j \in 1,2,... N$
3. $|S_i| = k \forall\ i \in 1, 2, 3... N$ where k<N. This is called equal work property as each site will send and receive equal number of messages for achieving ME.

4. Q_i is contained in k S_j's ∀ $i \in 1, 2, 3...N$. This is called equal responsibility property.

   N is the number of sites, Q_i refers to the i^{th} site of the communication network.

   S_i is a set of k Q_j's ∀ $i, j \in 1,2, \ldots N$

Condition 3 states that the size of the requests sets of all sites must be equal implying that all sites should have to do equal amount of work to invoke mutual exclusion. Condition 4 enforces that exactly the same number of sites should request permission from any site implying that all sites have "equal responsibility" in granting permission to other sites. Maekawa explained that for a fixed k, maximum possible value of N would be k(k-1)+1 with the assumption that any two quorums have only one intersection site. Hence the theoretical lower bound of quorum size is approximately √N. Theoretically quorums can be generated by trying all combinations of the requesting sites which satisfy the above properties.

Maekawa's algorithm can be stated as:

A site S_i executes the following steps to execute the CS.

Requesting the critical section

(a) A site S_i requests access to the CS by sending REQUEST (i) messages to all sites in its request set R_i .

(b) When a site S_j receives the REQUEST (i) message, it sends a REPLY(j) message to S_i provided it hasn't sent a REPLY message to a site since its receipt of the last RELEASE message. Otherwise, it queues up the REQUEST (i) for later consideration.

Executing the critical section

(c) Site S_i executes the CS only after it has received a REPLY message from every site in R_i

Releasing the critical section

(d) After the execution of the CS is over, site S_i sends a RELEASE (i) message to every site in Ri .

(e) When a site S_j receives a RELEASE (i) message from site S_i , it sends a REPLY message to the next site waiting in the queue and deletes that entry from the queue. If the queue is empty, then the site updates its state to reflect that it has not sent out any REPLY message since the receipt of the last RELEASE message.

Since the size of a request set is √N(which is the size of a quorum) , an execution of the CS requires √N REQUEST, √N REPLY, and √N RELEASE messages, resulting in 3√N messages per CS execution. Synchronization delay in this algorithm is 2T. This is because after a site *Si* exits the CS, it first releases all the sites in R_i and then one of those sites sends a REPLY message to the next site that executes the CS. Maekawa's algorithm can deadlock because a site is exclusively locked by other sites and requests are not prioritized by their timestamps. Assume three sites S_i, S_j, and S_k simultaneously invoke mutual exclusion. Suppose $R_i \cap R_j = \{S_{ij}\}$, $R_j \cap R_k = \{S_{jk}\}$, and $R_k \cap R_i = \{S_{ki}\}$. Consider the following scenario:

S_{ij} has been locked by S_i (forcing S_j to wait at S_{ij}).

S_{jk} has been locked by S_j (forcing S_k to wait at S_{jk}).

S_{ki} has been locked by S_k (forcing S_i to wait at S_{ki}).

This state represents a deadlock involving sites S_i, S_j, and S_k. Maekawa's algorithm handles deadlocks by requiring a site to yield a lock if the timestamp of its request is larger than the timestamp of some other request waiting for the same lock. A site suspects a deadlock (and initiates message exchanges to resolve it) whenever a higher priority request arrives and waits at a site because the site has sent a REPLY message to a lower priority request. Deadlock handling requires three types of messages:

FAILED: A FAILED message from site S_i to site S_j indicates that Si cannot grant S_j's request because it has currently granted permission to a site with a higher priority request.

INQUIRE: An INQUIRE message from S_i to S_j indicates that S_i would like to find out from S_j if it has succeeded in locking all the sites in its request set.

YIELD: A YIELD message from site S_i to S_j indicates that S_i is returning the permission to S_j (to yield to a higher priority request at S_j).

Maekawa's algorithm handles deadlocks as follows:

When a REQUEST(ts, i) from site S_i blocks at site S_j because S_j has currently granted permission to site S_k , then S_j sends a FAILED(j) message to S_i if S_i's request has lower priority. Otherwise, S_j sends an INQUIRE(j) message to site S_k . In response to an INQUIRE(j) message from site S_j, site S_k sends a YIELD(k) message to S_j provided S_k has received a FAILED message from a site in its request set or if it sent a YIELD to any of these sites, but has not received a new GRANT from it. In response to a YIELD(k) message from site S_k, site S_j assumes as if it has been released by S_k, places the request of S_k at appropriate location in the request queue, and sends a GRANT(j) to the top request's site in the queue. Maekawa's algorithm

requires extra messages to handle deadlocks Maximum number of messages required per CS execution in this case is 5√N.

Maekawa's original paper explains the construction of finite projective plane but not all projective planes exist. So Maekawa gave another algorithm called grid based algorithm which avoids the construction of finite projective planes. Here the sites are organized as a grid of squares. A quorum can be constructed by the union of row and column containing the requesting site. In this algorithm two sites S_i and S_j intersect each other in two sites $\forall i, j$. So any two quorums have two intersections. The quorum size is roughly twice the theoretical lower bound as proposed in the finite projective planes i.e. 2√N – 1. Though this algorithm is simple to understand, but any two quorums have two intersections here.

A better option has been suggested by Wai- Shing Luk and Tien- Tsin Wong is to construct a quorum using either a row or a column. Here N is no longer a perfect square. The sites can be organized in the form of a right angled triangle. Starting from the leftmost node on the first row, move farthest right horizontally along the row and take a 90 degree turn (when no more nodes exist) to the bottom along the column. The line joining such a row and column contains the sites of a quorum. All such quorums can be formed starting at different rows. This scheme is called the row based scheme.

Another similar scheme is the column based triangle configuration. Here a line is drawn starting from the rightmost and the bottommost node. This node takes a 90 degree turn to the left and covers the entire row (if there is no other node along its path in the column). Thus the line reaches the leftmost node staring from the farthest bottommost node. The nodes joined by this line constitute a quorum. Any two such lines meet at exactly one node. Therefore, any two quorums have exactly one

intersection (property 2) in both the row and the column based schemes. The size of the quorum is smaller near the top of the grid (only entire row included) in the row based scheme. Similarly, the size of the quorum is smaller at the bottom of the grid where only column of the sites is included. This causes a violation to the Property 4 which is the equal responsibility property. To solve this problem, a combination of row based and column based schemes is used. Here the requesting site does not have to tell other sites about whether it is using a row based scheme for quorum construction or the column based one. The quorum size is approximately $\sqrt{2N}$.

Another method to construct quorums assumes that the cellular system is organized as a binary tree. The quorum consists of a branch from the root node to the leaf node. This method is called the tree-quorum algorithm and has been given by Agrawala and Abbadi. The size of the quorum is $\sqrt{N}$ here. All the sites in the system are logically organized into a complete binary tree. For a complete binary tree with level 'k', we have 2k+1 – 1 sites with its root at level k and leaves at level 0. The number of sites in a path from the root to a leaf is equal to the level of the tree k+1 which is equal to O (log n). A path in a binary tree is the sequence a_1, a_2. . . a_i, a_{i+1}. . . . a_k such that ai is the parent of a_{i+1}. The algorithm tries to construct quorums in a way that each quorum represents any path from the root to a leaf. If it fails to find such a path (say, because node 'x' has failed), the failed node 'x' is substituted by two paths both of which start with the left and right children of 'x' and end at leaf nodes. If the leaf site is down or inaccessible due to any reason, then the quorum cannot be formed and the algorithm terminates with an error condition. The sets that are constructed using this algorithm are termed as tree quorums. *When there is no node failure, the number of quorums formed is equal to the number of leaf sites.*

Consider the tree of height 3, constructed from 15 sites. In this case 8 quorums are formed from 8 possible root-leaf paths: 1-2-4-8, 1-2-4-9, 1-2-5-10, 1-2-5-11, 1-3-6-12, 1-3-6-13, 1-3-7-14 and 1-3-7-15. If any site fails, the algorithm substitutes for that site two possible paths starting from the site's two children and ending in leaf nodes. For example, when node 3 fails, we consider possible paths starting from children 6 and 7 and ending at leaf nodes. The possible paths starting from child 6 are 6-12 and 6-13, and from child 7 are 7-14 and 7-15. So, when node 3 fails, the eight quorums can be formed: {1,6,12,7,14}, {1,6,12,7,15}, {1,6,13,7,14}, {1,6,13,7,15}, {1,2,4,8}, {1,2,4,9},{1,2,5,10}, {1,2,5,11}.

Since the number of nodes from root to leaf in an 'n' node complete tree is log n, the best case for quorum formation, i.e., the least number of nodes needed for a quorum is log n. When the number of node failures is greater than or equal to log n, the algorithm may not be able to form tree-structured quorum. So, as long as the number of site failures is less than log n, the tree quorum algorithm guarantees the formation of a quorum and it exhibits the property of 'graceful degradation'.

Another algorithm for constructing quorums is given by Agrawala and Abbadi using a modified grid. The quorums constructed are called billiard quorums. Instead of using horizontal and vertical line segments of rows and columns in the grid scheme of Maekawa, they used paths that resemble billiard paths. This reduced quorum size to $\sqrt{2}N$. However, the size of each quorum has to be an odd integer, and the method does not satisfy the equal responsibility property of Maekawa. The features of billiard's algorithm are:

- Assumes system nodes arranged into a 2-D grid.
- Billiard Quorums are of size $2^{1/2} * N^{1/2}$.

- Assumes number of nodes N= $(q^2-1)/2$ and assumes that q is odd (maintain symmetry).
- Let P be located in cell(i,j), and denoted by R(i,j) then its quorum members can be found along the lines:

  R (i-1, j+j) = R (i,j)-(q-1)/2

  R (i+1, j+1) = R (i,j)+(q+1)/2

  R (i+1, j-1) = R (i,j)+(q-1)/2

  R (i-1, j-1) = R (i,j)-(q+1)/2

CHAPTER 4: CELLULAR NETWORKS

4.1 About cellular networks

A cellular communication network consists of clusters of hexagonal cells each having a fixed base station called mobile service station (MSS). All the MSS's are connected to each other through a fixed communication network.

The frequency spectrum allocated to wireless communications is very limited, so the cellular concept was introduced to reuse the frequency. Each cell is assigned a certain number of channels. To avoid radio interference, the channels assigned to one cell must be different from the channels assigned to its neighbouring cells. However, the same channels can be reused by two cells that are far apart such that the radio interference between them is tolerable. By reducing the size of cells, the cellular network is able to increase its capacity, and therefore to serve more subscribers.

Some channels are forward (or downlink) channels that are used to carry traffic from the base station to mobile stations, and the others are reverse (or uplink) channels that are used to carry traffic from mobile stations to the base station. Both forward and reverse channels are further divided into control and voice (or data) channels. The voice channels are for actual conversations, whereas the control channels are used to help set up conversations.

A mobile station communicates with another station, either mobile or land, via a base station. A mobile station cannot communicate with another mobile station directly. To make a call from a mobile station, the mobile station first needs to make

a request using a reverse control channel of the current cell. If the request is granted by the MSC, a pair of voice channels will be assigned for the call. To route a call to a mobile station is more complicated. The network first needs to know the MSC and the cell in which the mobile station is currently located. How to find out the current residing cell of a mobile station is an issue of location management. Once the MSC knows the cell of the mobile station, it can assign a pair of voice channels in that cell for the call. If a call is in progress when the mobile station moves into a neighbouring cell, the mobile station needs to get a new pair of voice channels in the neighbouring cell from the MSC so the call can continue. This process is called handoff (or handover). The MSC usually adopts a channel assignment strategy that prioritizes handoff calls over new calls.

The use of particular frequency in a cell for communication session establishment can be viewed as equivalent to entering the critical section by the cell in which the channel is being used. In mobile communication, frequency channels are the common resource. Each cell has to attain a frequency channel for communication. Since two or more neighbouring cells can try to attain the same channel at the same time, this can be viewed as ***similar to Mutual Exclusion Problem*** where the processes wait for a shared resource currently being used by some other process in order to complete their task.

The major factor that determines the number of channels with a predetermined quality is the level of received signal quality that can be achieved in each channel. This level strongly depends on the interference effects. Some possible sources of interference may be another carrier in the same cell, a call in progress in a neighboring cell, other base stations operating in the same frequency band, or any non cellular system that radiates in the same frequency band. Interference on voice

channels causes crosstalk—the subscriber hears interference in the background due to another call. On control channels, interference leads to missed calls and blocked calls. Interference is more severe in urban areas, due to industrial interference and a large number of base stations and mobiles in the proximity. It has been recognized as a major bottleneck in increasing capacity. Interference to a channel that serves a particular call occurs mainly when a user in an adjacent cell uses the same channel (co channel interference), a user in the same region uses an adjacent channel (co site interference), or a user in an adjacent region uses an adjacent channel (adjacent channel interference).

4 .2 Co channel Interference

Frequency reuse increases the system's spectrum efficiency, but interference due to the common use of the same channel may occur if the system is not properly planned. This kind of interference is called co channel interference. Co channel interference is the most critical of all interferences that occur in cellular radio; it depends on cellular traffic. The possibility of co channel interference appearing is greater in the busy hours of a cellular system. The total suppression of co channel interference is achieved by not using the frequency reuse concept, which is contradictory to the whole idea of the cellular radio. Thus, in order to obtain a tolerable value of co channel interference, the system planner has to take into account the reuse distance D. When the size of each cell in a cellular system is roughly the same, co channel interference is independent of the transmitted power and becomes a function of the radius of the cell R and the reuse distance D. The factor $Q = \sqrt{3}\ K$ is called the co channel interference reduction factor or reuse factor and is the measure of co channel interference. The Q factor determines spectral

efficiency within a cell and is related to the number of cells in the cluster *K*. Assuming that all the cells transmit the same power, the frequency reuse distance *D* can be increased by increasing *K*. One could expect that by making *K* as large as possible, all problems concerning co channel interference could be solved. An advantage of large clusters is the fact that the interference from co channel cells decreases because the distance between the co channel cells also increases with the increase in cluster size. On the other hand, the available bandwidth and therefore the available number of channels is fixed. When *K* is large, the number of channels per cell is small. That causes spectrum inefficiency.

4.3 Co site and Adjacent Channel Interference

In addition to co channel interference, a second source of noise is the interference between two adjacent channels of the same (co site interference) or adjacent cells (adjacent channel interference). It should be noted that the adjacent channel here is not the close neighbouring channel in a strict communication sense, but rather the nearest assigned channel in the same cell and can be several channels apart. Co site and adjacent channel interference result from equipment limitations, mainly from imperfect receiver filters that allow nearby frequencies to leak into the pass band. The problem can be particularly serious if one adjacent channel user is transmitting in close range to a receiver that is attempting to receive a weaker signal using a neighbouring channel. Several techniques can be used in order to solve this problem. The total frequency spectrum is usually split into two halves so that the reverse channels that compose the up-link (mobile to base station) and the forward channels that compose the down-link (base station to mobile) can be separated by half of the spectrum. If other services can be inserted between the two halves, then a greater frequency separation can be attained. Co site and adjacent

channel interference can also be minimized through careful channel assignments. By keeping the frequency separation between each channel in a given cell as large as possible, these types of interference may be reduced considerably. Some designers also prevent a source of adjacent channel interference by avoiding the use of adjacent channels in geographically adjacent cell sites. This strategy, however, is dependent on the cellular pattern. For instance, if a seven-cell cluster is chosen, adjacent channels are inevitably assigned to adjacent cells.

4.4 Frequency management and channel assignment issues:

Efficient spectrum resource management is of paramount importance due to increasing demands of new services, rapid and unbalanced growth of radio traffic, and other factors. A given radio spectrum (bandwidth) dedicated for cellular communications can be divided into a set of disjoint and non interfering radio channels. Techniques such as frequency, time, and code division can be used in order to divide the radio spectrum. In frequency division, the spectrum is divided into frequency bands. In time division, the usage of the channel is divided into time slots that are disjoint time periods. Finally, in code division, the channel separation is achieved by using different modulation codes. Moreover, other techniques based on the combination of the above methods can be used. Since the radio spectrum is finite in mobile radio systems, the most significant challenge is to use the radio-frequency spectrum as efficiently as possible. Geographic location is an important factor in the application of the frequency reuse concept in mobile cellular technology to increase spectrum efficiency. The techniques for increasing the frequency spectrum can be classified as:

Increase the number of radio channels

Improve spatial frequency spectrum reuse

Use proper frequency management and channel assignment techniques

Improve spectrum efficiency in time

Reduce the load of invalid calls (call forwarding, queuing, etc.)

The function of frequency management is to divide the total number of available channels into subsets that can be assigned to each cell either in a fixed fashion or dynamically. The terms frequency management and channel assignment are often confused. Frequency management refers to designating set-up channels and voice channels, numbering the channels, and grouping the voice channels into subsets (done by each system according to its preference). Channel assignment has to do with the allocation of specific channels to cell sites and mobile units. A fixed channel set that consists of one or more subsets is assigned to a cell site on a long-term basis. During a call, a specific channel is assigned to a mobile unit on a short-term basis. Frequency planning is therefore one of the most challenging tasks in designing a cellular mobile network. An accurate radio planning tool is essential for calculating predicted signal strength coverage and interference levels and satisfying the overall grade of service. The allocation of frequency channels to cells in a cellular network is a critical element of the design process since it affects the two major metrics of any cellular network: capacity and quality of service. The basic input data of a good frequency planning algorithm are the numbers of required channels for each cell and interference probabilities between each pair of cells using the same (co channel interference) or adjacent channels (adjacent channel interference) of a certain band. This data is usually provided by measurements or by simulation of radio wave propagation in the areas of interest. Different benefit criteria should be

taken into account when allocating channels to base stations. First of all, the interference between each pair of cells must not exceed a certain maximum threshold. This can be expressed using a proper compatibility matrix, which is a squared matrix that has as many rows or columns as cells in the system. The element values of the matrix represent the minimum allowable distance between channels in two cells. Channels should be allocated as to satisfy all traffic requirements per cell while observing the compatibility constraints. The assumptions regarding interference require the use of a large margin in the minimum acceptable signal-to-interference ratio in order to cope with the variations in the desired received and interference signals on both links. These signal variations are basically due to:

_ *Propagation conditions,* due to path loss and fading appearance.

_ *User mobility*—when the mobile approaches the cell boundary, the co channel interference at the mobile increases.

_ *Traffic load*—if more users share the same channel, co channel interference in the system increases.

4.5 Channel assignment

Channel assignment is a fundamental task of resource management that increases the fidelity, capacity, and quality of service of cellular systems by assigning the required number of channels to each cellular region in such a way that both efficient frequency spectrum utilization is provided and interference effects are eliminated. The channel allocation strategy can be seen as a method of assigning available channels to calls originating in the cells. If the strategy is unable to assign a channel, the call is blocked. The basic goal to be achieved by channel allocation techniques under the prism of the rapidly growing demand for cellular mobile services is to efficiently utilize the available spectrum so as to achieve optimum

system performance. The main focus on research concerning channel assignment is to find strategies that give maximal channel reuse without violating the constraints so that blocking is minimal. Constraints can be classified into three categories:

1. The frequency constraint specifies the number of available frequencies (channels) in the radio spectrum. This constraint is imposed by national and international regulations.

2. The traffic constraints specify the minimum number of frequencies required by each station to serve a geographic area. These constraints are empirically determined by the telecommunications operators.

3. The interference constraints are further classified as:

The co channel constraint—the same channel cannot be assigned to certain pairs of radio cells simultaneously.

The adjacent channel constraint—frequencies adjacent in the frequency domain cannot be assigned to adjacent radio cells simultaneously.

The co site constraint—any pair of channels assigned to a radio cell must occupy a certain distance in the frequency domain.

Constraints in the frequency assignment problem are therefore multiple and some of them are conflicting. The most severe limitation is the frequency constraint. This constraint imposes a high degree of frequency reuse by the stations and consequently increases the difficulty of satisfying the interference constraints. Most channel assignment schemes are quite detailed and founded largely on ad-hoc principles. Moreover the channel assignment schemes are evaluated using different benchmarks following extended simulations with a variety of assumptions regarding the mobile radio environment. Some of these assumptions might be the cellular topology, the different choice of reuse factors, the use of different traffic patterns, the

incorporation of propagation factors, the use of mobility, etc. The combination of these factors makes a systematic comparison of the various channel allocation methods quite infeasible and a true decision of the best scheme is difficult to attain. Roughly speaking, channel assignment is generally classified into fixed and dynamic assignment. In fixed channel assignment (FCA), channels are nominally assigned to cells in advance according to the predetermined estimated traffic intensity.

In dynamic channel assignment (DCA), channels are assigned dynamically as calls arrive. The latter method makes cellular systems more efficient, particularly if the traffic distribution is unknown or changes with time, but has the disadvantage of requiring more complex control and is generally time consuming.

Various extensions or combinations of the above two schemes have been discussed in the literature. The most basic ones are hybrid channel assignment (HCA) and borrowing channel assignment (BCA).

In HCA, the set of the channels of the cellular system is divided into two subsets; one uses FCA and the other DCA.

In the BCA scheme, the channel assignment is initially fixed. If there are incoming calls for a cell whose channels are all occupied, the cell borrows channels from its neighbouring cells and thus call blocking is prevented.

FCA is the simplest off-line allocation scheme. It has been used as the primary allocation technique for first- and second-generation cellular systems and outperforms DCA and other schemes under uniform and heavy traffic loads. Moreover FCA problems can serve as bounds for the performance of HCA and DCA schemes. For these reasons, FCA constitutes a significant research subject for the operations research, artificial intelligence, and mobile communication fields.

4.6 Reuse distance

The main challenge, the mobile technology is facing today is the effective utilization of bandwidth as the number of mobile users are increasing at a very fast speed. To enable large number of users to communicate efficiently without call blocks/drops, frequencies can be reused between cells separated by a minimum reuse distance D_{min}.

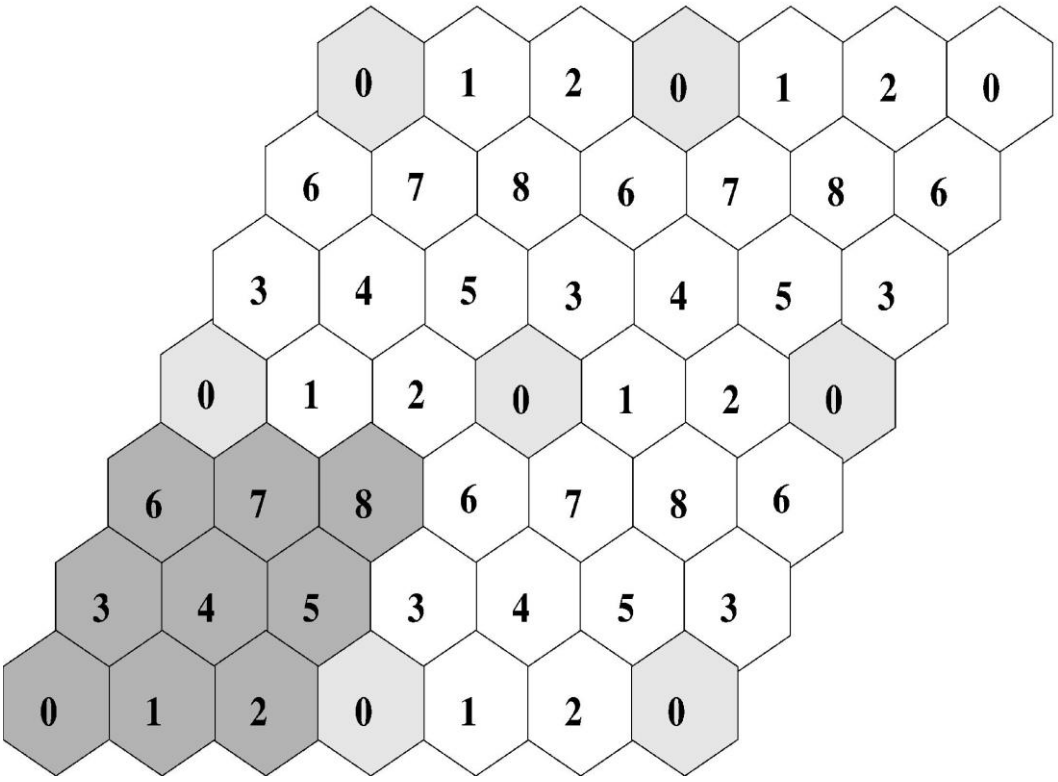

Figure 1: A hexagonal cellular network

An x*y cellular network has x rows and y columns of cells. Cell at i^{th} row and j^{th} column is denoted as (i, j). The distance between the cells is defined as the Euclidean distance between the centres of two cells. The distance between any two cells $C_1 = (i_1, j_1)$ and $C_2 = (i_2, j_2)$ is

Dist (C_1, C_2) =$\sqrt{}$ $(i_1-i_2)^2 + (i_1-i_2)(j_1-j_2) + (j_1-j_2)^2$ (eqn. 1)

Let the centres of two cells be (0, 0) and (a, b). Then distance between them

D_{min} =$\sqrt{}$ $(a^2 + a*b + b^2)$ using eqn. 1

This distance is called the minimum reuse distance. The nearest co channel cells to a cell are those cells whose separating distance is exactly D_{min}. The distance between two cells is defined as the distance between the centres of the cells. A channel can be reused in any two cells if the distance between them is at least $D_{min.}$ This means that two channels having a separation of greater than or equal to D_{min} can use the same frequency channel at the same time without any interference in order to achieve good quality communication between the mobile users. This enhances the use of allotted bandwidth. Two cells having a separating distance less than D_{min} cannot use the same frequency as their calls would interfere with each other. So adjacent cells are not allotted same frequencies.

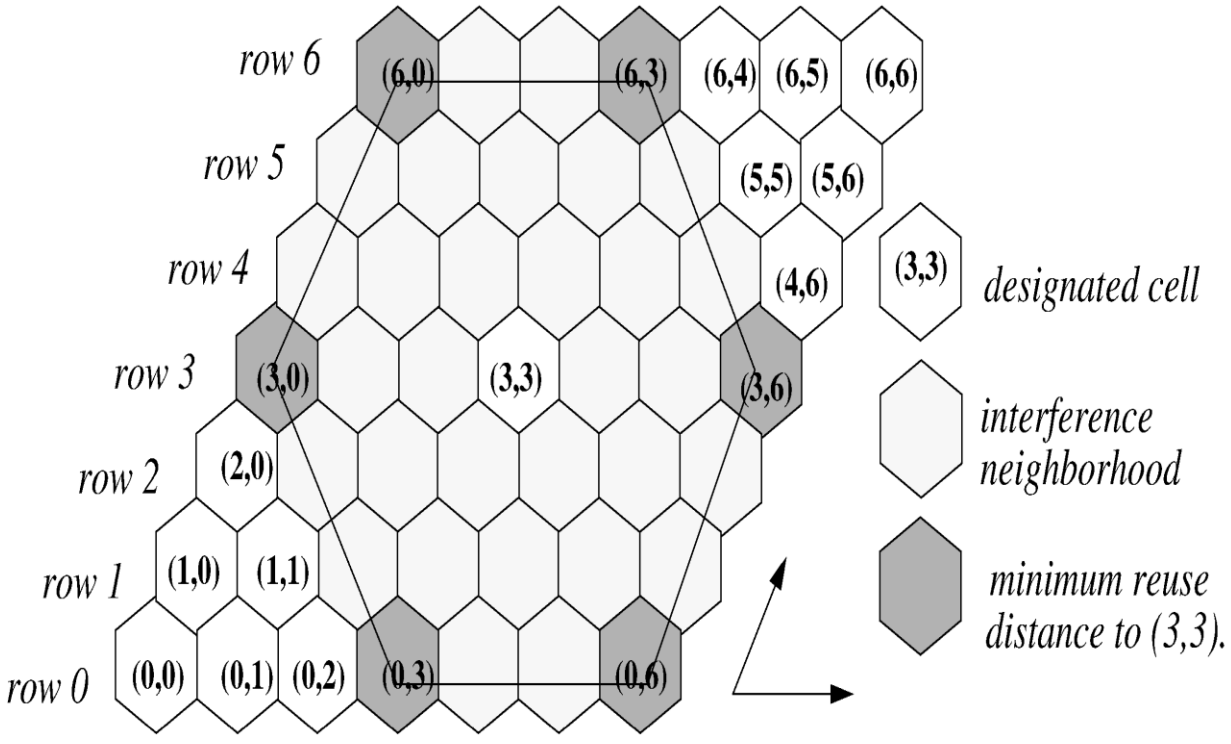

Figure 2. ACellular network With D_{min} =3, N =9.

Frequency reuse leads to the idea of relaxed mutual exclusion since multiple distinct critical sections can be executed concurrently in the cells separated by distance greater than or equal to D_{min}. ME can be considered as a special case of RME. In ME two processes cannot use the same resource at the same time.

The bandwidth assignment in a mobile cellular system can be static, dynamic or the hybrid of the two. Static channel allocation means fixed channels are permanently assigned to each cell in order to handle the calls. When a MH requests for a call establishment, free channels in the allotted band are searched. If such a channel is available, call can be established. If no free channel is available in the cell, the call is dropped. Such a scheme cannot handle the increasing traffic load. So dynamic channel allocation is used where channels can move between the cells according to the traffic load in the cells. As cells can use each other's free channels, each cell must maintain a list of its own available channels and also that of its

neighbours. In order to maintain such huge database for implementing dynamic channel allocation, either centralized or distributed dynamic channel allocation schemes are used. In centralized dynamic channel allocation, the mobile switching centre (MSC) is the centralized authority which contains all information about available channels in the mobile system. It assigns the channels from this pool and later on when the channels are released after usage, they are returned back to this common pool of available channels. As centralized scheme suffers from a single point of failure at the MSC, distributed dynamic channel allocation (DDCA) schemes are used. In DDCA each cell maintains lists of available and busy channels of itself

and its neighbours. When a call needs to be established, a channel is searched from these lists. If a free channel is available, it is assigned to that call. This scheme is called DDCA since message passing is used to exchange the status of channels at any instant among various cells. In hybrid scheme, some channels in a cell are permanently allotted while others are dynamically assigned.

CHAPTER 5: WORK DONE

A number of permission based DME algorithms were discussed in the last chapter. Different algorithms show different performance capabilities with respect to different metrics. These algorithms can be evaluated on the basis of performance measures like message complexity, synchronization delay, availability, communication delay, etc. An algorithm may be optimal with respect to one of these parameters but may show poor performance as regards another.

Network topology plays an important role in selecting the mutual exclusion technique to be used. Actual distances between sites in a network may also be important as suggested by Fu. Thus, it is not possible to single out an algorithm as the best algorithm for achieving mutual exclusion. A number of algorithms are available from which one can choose the one which best suits the application in hand, keeping an eye on the system requirements, network topology and the performance measure that needs to be optimized. The main drawback of permission based mutual exclusion algorithms is that the communication cost to enter critical section is directly proportional to the size of quorums. Much research has been done to optimize communication and minimize the size of quorums.

I have tried to provide a comparative study of following three algorithms, bringing out their common features and listing out their differences on the basis of performance metrics like message complexity, synchronization delay. For this the implementation of these algorithms was carried out in Matlab.

1. Maekawa's Grid algorithm
2. Luk Wong's row and column scheme

3. Agrawala and Abbadi Billiard's algorithm

Performance Metrics

The performance is generally measured by the following four metrics:

Message complexity: The number of messages required per CS execution by a site.

Synchronization delay: After a site leaves the CS, it is the time required and before the next site enters the CS (see Figure 3).

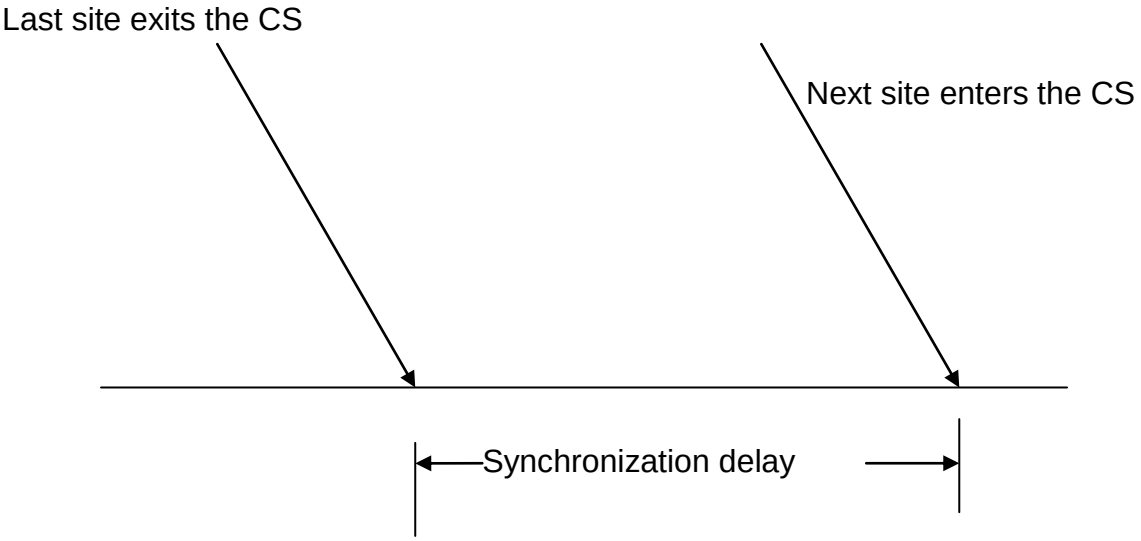

Figure: 3 Synchronization Delay

The delay from the moment a process needs to enter a critical region until its actual entry also varies. When critical regions are short and rarely used, the dominant factor in the delay is the actual mechanism for entering a critical region. When they are long and frequently used, the dominant factor is waiting for everyone else to take their turn.

Response time: The time interval a request waits for its CS execution to be over after its request messages have been sent out.

System throughput: The rate at which the system executes requests for the CS. system throughput=1/(SD+E) where SD is the synchronization delay and E is the average critical section execution time.

Lamport presented the first lock-based solution. The mutual exclusion algorithm applies to fully connected networks with in-order delivery of messages. It has a message complexity of 3*(*N*-1), where *N* is the number of processes.

Ricart and Agrawala have sought to improve upon the message complexity to 2*(*N*-1). They achieve the reduction by reducing the number of messages exchanged between a requesting site and participating sites. Rather than sending a REPLY immediately after receiving a REQUEST, as in Lamport's algorithm, a process will only do so when it is ready to grant its critical section lock to the requestor. This selective use of REPLY eliminates the need for the RELEASE message. Figure 4 gives the comparison of Lamport's and Ricart and Agrawala's algorithms of mutual exclusion on the basis of message complexity.

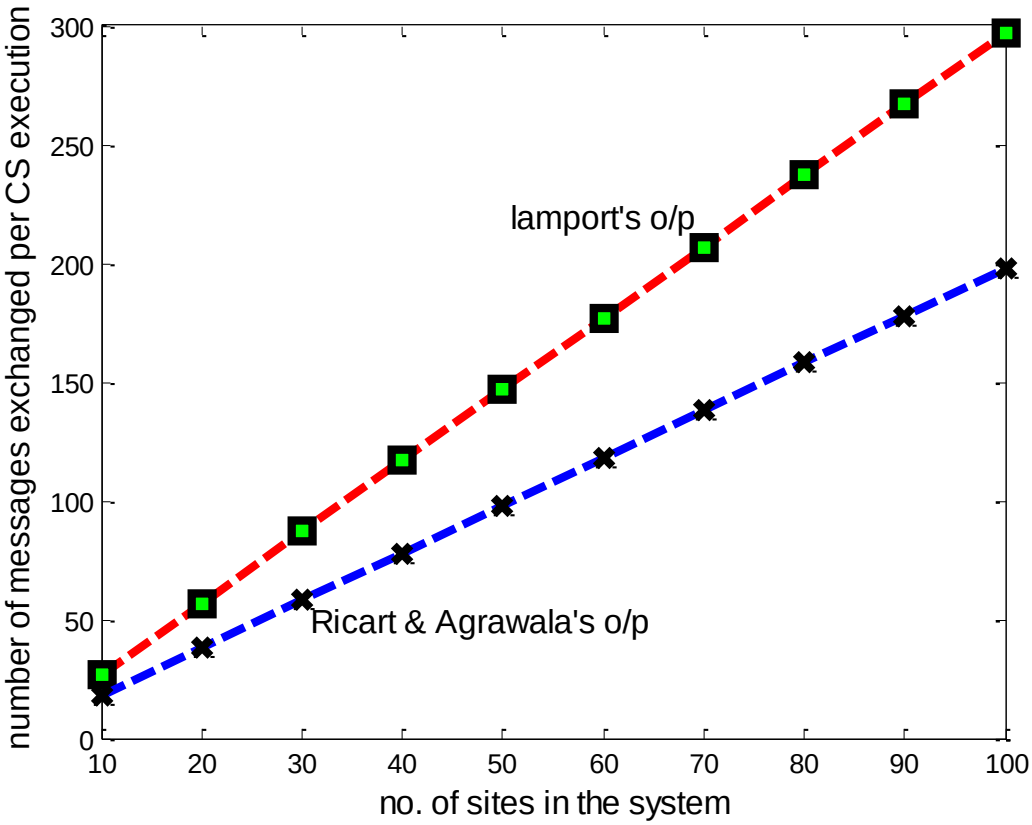

Figure 4 Comparison of Message complexity of the system

Instead of reducing the number of messages per process pair, Maekawa reduces the number of processes that must be contacted in order to assure mutual exclusion. Each process possesses a set of neighbours, referred to as its quorum. Comparison of the message complexity of the Ricart-Agrawala, Lamport and Maekawa algorithm is done. The following hypotheses formally state the expected outcomes:

1. For a fixed load, *M* in Ricart-Agrawala, Lamport varies linearly with *N*.
2. For a fixed load, *M* in Maekawa varies according to $\sqrt{N}$.

Figure 5 below shows the comparison of Ricart-Agrawala, Lamport and Maekawa algorithm on the basis of message complexity.

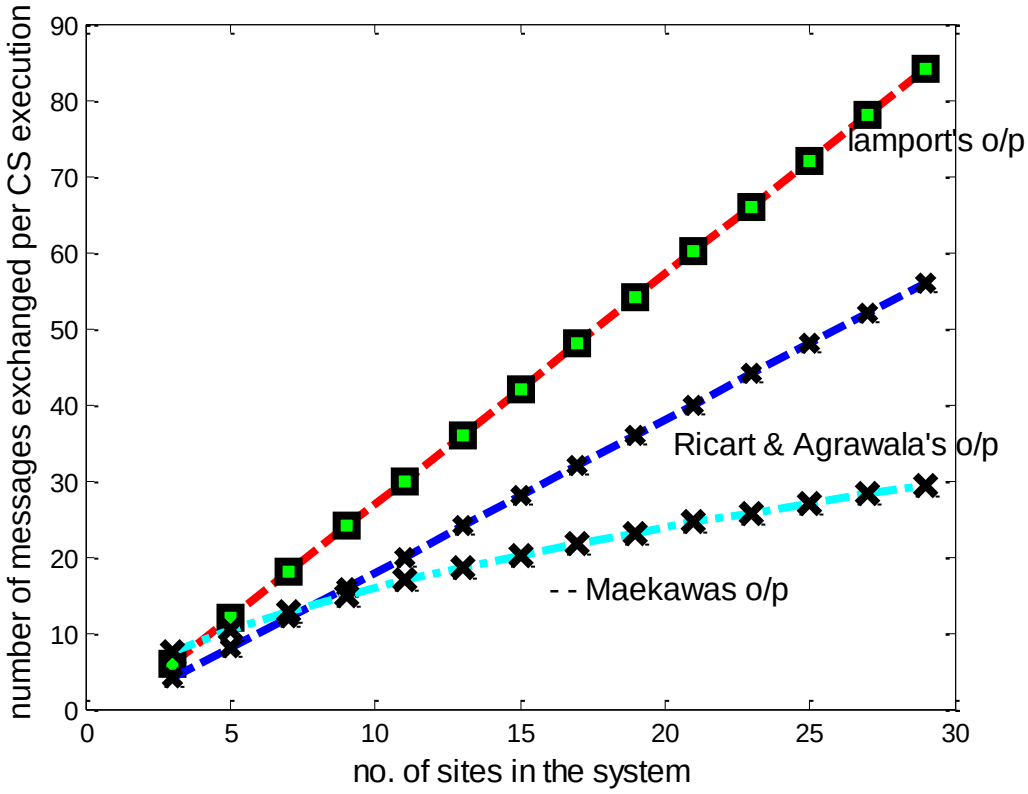

Figure 5: Comparison of Message complexity of permission based algorithms

In Maekawa's algorithm, the quorum size is 2√N. In Luk Wong's and Agrawala & Abbadi algorithms, the quorum size is √2N where N is the number of sites in the system. The algorithms implemented in Matlab suggest the same. The figure 6 shows the comparison of various quorum based algorithms based on message complexity. Message complexity increases as the size of quorum's increases.

Maekawa's square grid algorithm has the highest message complexity among the above discussed quorum based algorithms.

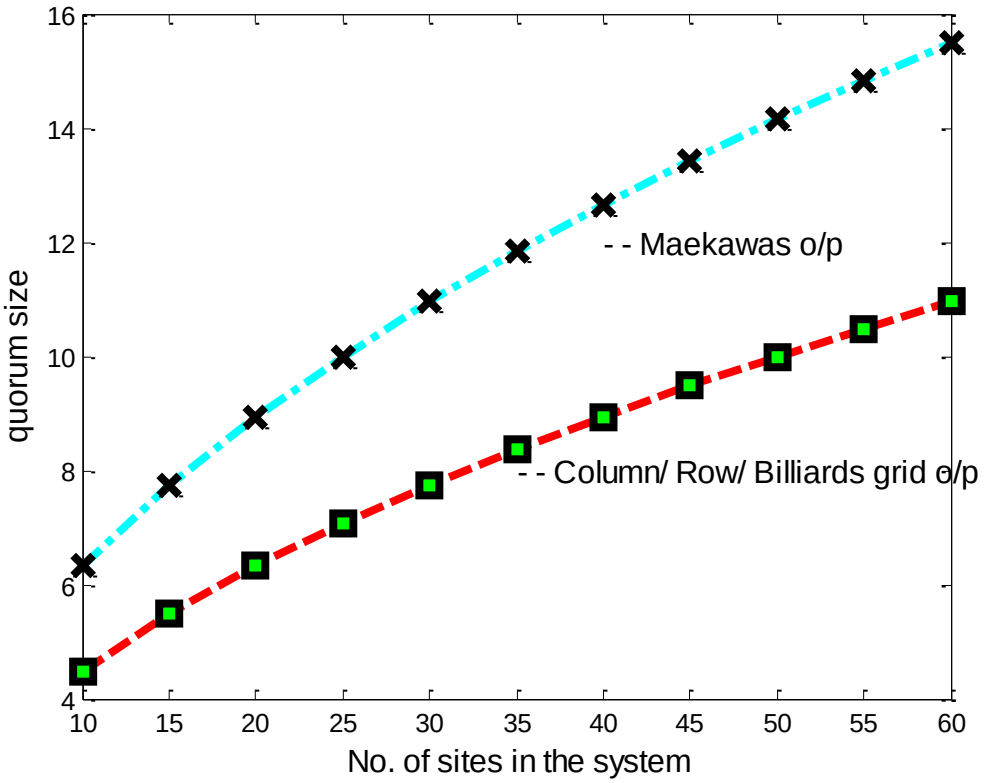

Figure 6: Comparison of message complexity of quorum based algorithms

Synchronization delay in the Lamport and Ricart and Agrawala algorithm is T. Synchronization delay in Maekawa's, Luk Wong and Billiards algorithms is 2T. This is because after a site Si exits the CS, it first releases all the sites in Ri and then one of those sites sends a REPLY message to the next site that executes the CS.

Quora have the following properties: The quora of any two processes in the system have a non-zero intersection, and a process belongs to its own quorum. A process will only grant its critical section lock to one member of its quorum at a time. Consequently, if a process succeeds in acquiring locks from its entire quorum, it has guaranteed mutual exclusion for the entire system.

The properties of quorums namely mutual exclusion, equal effort, equal responsibility have been validated in the following algorithms using Matlab

1. Maekawa's Grid algorithm
2. Luk Wong's row and column scheme
3. Agrawala and Abbadi Billiard's algorithm

Luk Wong's row and column scheme does not support the equal responsibility property as does the Billiard's algorithm.

CHAPTER 6: CONCLUSION & FUTURE WORK

6.1 Conclusion

Study was carried out on various Quorum based algorithms to compare message complexity of the system. It was concluded that the message complexity of the system depends on the size of quorums constructed using the various algorithms studied under the thesis. It increases as the quorum size increases which increases the communication cost of the system. Also it was concluded that Luk Wong and Agrawala and Abbadi algorithms to construct quorums do not follow Maekawa's equal responsibility property. The synchronization delay is more in quorum based algorithms than the lock based permission algorithms as the release message needs to be sent to the quorum members by a site who has just finished using a channel (shared resource)

6.2 Future Work

The tree based algorithm of Agrawala and Abbadi is proposed to be studied and implemented. The performance metrics viz., message complexity, synchronization delay of the tree based Agrawala and Abbadi algorithm would be compared with the algorithms implemented in the thesis.

NS-2 network simulator will be studied and a cellular network will be simulated in this tool to study the performance metrics.

REFERENCES

[1] Ravi Prakash, Niranjan, G. Shivaratri, Mukesh Singhal, “Distributed Dynamic Channel Allocation for Mobile Computing”, 1995 ACM.

[2] Jianping Jiang, “On Distributed Dynamic Channel Allocation in Mobile Cellular Networks”, IEEE transactions on Parallel And Distributed Systems, vol. 13, No. 10, 2002.

[3] P.C.Saxena, J.Rai, “A survey of permission- based distributed mutual exclusion algorithms”, Computer Standards & Interfaces 25 (2003) 159–181.

[4] W. Luk, T. Wong, Two new quorum based algorithms for distributed mutual exclusion, Proceedings of the 17th International Conference on Distributed Computing Systems, ICDCS’ 97, Baltimore, MD, USA, IEEE (1997, May),pp. 100–106.

[5] D.J Goodman, Wireless Personal Comm. Systems, Addison Wesley

[6] M. Maekawa, “A √N algorithm for mutual exclusion in decentralized systems”, ACM Trans. Comput. Systs., pp 145-159, May 1985

[7] W.C.Y.Lee, Mobile Cellular Telecommunications: Analog and Digital Systems. McGraw – Hill

[8] V.H.MacDonald, “The Cellular Concept”,The Bell System Technical J.., vol. 58, no. 1

[9] Ricart G and Agarwala, A. K. “An optimal algorithm for mutual exclusion in computer networks”, CACM., vol. 24, no.1, 1981

[10] Kumar, A. " Hierarchical Quorum Consensus: a new algorithm for managing replicated data," IEEE Trans. Comp., vol. 40, no. 9, 1991

[11] Shing- Tsaan Huang, Jehu- Ruey Jiang and Yu- Chen- Kuo, National Tsing Hua university, Hsin Chu, Tiawaan , " k-coteries foe Fault- Tolerant k entries to a Critical Section", NSC Republic of China.

[12] Gifford, D.K. "Weight voting for replicated data", in Proc. 7th ACM SIGOPS Symp. Oper. Syst. Principles, Pacific Grove, CA, pp. 150- 159 , 1979

[13] Thomas, R.H, "A majority consensus approach to concurrency control," ACM Trans. Database Syst., vol 4, no. 2, pp. 180-209,1979

[14] Lamport, L. "Time, clocks and the ordering of events in a distributed system, CACM., vol. 21, no. 7, pp. 145-159, July 1978

[15] Agrawala, D., and El Abbadi, "An efficient and fault- tolerant algorithm for distributed mutual exclusion," ACM Trans. Comp. Syst., vol.9, no. 1, pp. 1-20, Feb.1991

[16] Handbook of Wireless Networks and Mobile Computing, Edited by Ivan Stojmenovic´Copyright © 2002 John Wiley & Sons, Inc.

[17] D. Agrawal,O¨ . Eg˘eciog˘lu, A.El. Abbadi, Billiard quorums on the grid, Information Processing Letters 64 (1997) 9– 16.

[18] H. Garcia-Molina, D. Barbara, "How to assign votes in a distributed system", Journal for the Association for Computing Machinery, 1985.

[19] A. W. Fu, “Delay-Optimal Quorum Consensus for Distributed Systems”, IEEE Trans. Parallel and Distributed Systems, vol. 8, no. 6, Jan. 1997.

[20] Megna Gupta, A.K Sachan, Journal of Theoritical and Applied Information Technology, 2007, “Distributed Dynamic Channel Allocation Algorithm for Cellular Mobile Network”.

[21] Jiangchang Yang et.al. “A fault tolerant channel allocation algorithm for cellular network with mobile base stations”. IEEE transactions on vehicular technology 56(1): 349- 361, 2007.

[22] A. Silberschatz, P. Galvin, Operating System Concepts, 5th ed., Addison-Wesley, Harlow, England, 1998.