On Distributed Dynamic Channel Allocation in Mobile Cellular Networks

Jianping Jiang, Ten-Hwang Lai, Neelam Soundarajan
Department of Computer and Information Science
The Ohio State University
Columbus, OH 43210, USA
Email: jjcmh@yahoo.com, {lai, neelam}@cis.ohio-state.edu

March 14, 2002

Abstract: Distributed dynamic channel allocation (DDCA) is a fundamental resource management problem in mobile cellular networks. It has a flavor of distributed mutual exclusion but is not exactly a mutual exclusion problem (because a channel may be reused in different cells). In this paper, we establish the exact relationship between the two problems. Specifically, we introduce the problem of relaxed mutual exclusion to model one important aspect of the DDCA problem. We develop a general algorithm that guarantees relaxed mutual exclusion for a single resource and prove necessary and sufficient conditions for the information structure. Considering distributed dynamic channel allocation as a special case of relaxed mutual exclusion, we apply and extend the algorithm to further address the issues that arise in distributed channel allocation such as deadlock resolution, dealing with multiple channels, design of efficient information structures, and channel selection strategies. Based on these results, we propose an example distributed channel allocation scheme using one of the information structures proposed. Analysis and simulation results are provided and show that the results of this research can be used to design more efficient distributed channel allocation algorithms.

Key Words: mutual exclusion, dynamic channel allocation, cellular networks, distributed algorithms.

1 Introduction

Channel allocation is a key problem in mobile cellular networks. In such a network, the geographical area is divided into hexagonal *cells* [7, 13]. The mobile telephones in each cell are serviced by a *base station* (BS) located at the center of the cell; the BSs are interconnected via a wired network. The available wireless bandwidth is divided into *channels*, where each channel is capable of supporting a communication session. A given channel can be used simultaneously in different cells without unacceptable interference provided the distance between each pair of cells is greater than or equal to the *minimum reuse distance* [15]. The problem of *channel allocation* is to allocate channels to the various cells so that the available bandwidth is most effectively used by the various cells to meet the traffic demand in each cell without interfering with calls in neighboring cells.

A number of different approaches have been used to address this problem. One possibility is to allocate the channels in a *static* manner, i.e., each cell is permanently allocated a fixed set of channels that it uses to handle all calls within that cell. One approach of this kind [16, 25, 27, 22] is to map the channel allocation problem to the *graph multicoloring* problem with the nodes of the graph representing the cells and the edges representing the constraints on which pairs of cells may (simultaneously) use which channels. The resulting channel allocation schemes are appropriate, even close to optimal, if the traffic pattern in the mobile network is *static*.

Our interest in this paper is to consider the problem of *dynamic* channel allocation. Whereas in static channel allocation, a set of channels is assigned permanently for use by a given cell, we want to be able, as traffic patterns change, to move channels from cells with less demand to those with heavier demand. The traffic pattern in a cellular network may change for a variety of reasons, from heavy demand in business districts during working hours, to increased demands in outlying suburban areas during evening hours at the same time that traffic in business districts falls off, to heavy demands around a sport arena on game days, or even a sudden increase in the volume of calls in a given cell because of an emergency such as a fire in the area served by the cell. Whatever the reasons for the changes, being able to move channels among cells to meet current demands clearly makes it possible to use the bandwidth more effectively than is possible with a static allocation. The tradeoff, of course, is the additional computation that needs to be done in keeping track of the changing traffic patterns, and in deciding which channels are to be moved from which cells to which other cells.

Centralized dynamic channel allocation has a long history [11]. Here, a single central pool of all available channels is maintained from which a central computer allocates channels to various cells on demand, and the cells return the channels to the central pool as the traffic in the respective cells go down. Various strategies that the central computer can adopt in choosing the particular channel to be allocated dynamically to particular cells, have been proposed; given these strategies, centralized allocation schemes can be quite effective in maximizing channel usage. But such schemes have a high centralization overhead, as well as a single point of failure at the central computer; both of these are serious concerns when the number of cells is large. Thus in this paper, we will consider *distributed* schemes for dynamic channel allocation. Each cell will interact with appropriate other cells to determine, in a mutually consistent fashion, exactly which channels to exchange between each other, based on the demand in each cell.

Some work has been done on the problem of distributed dynamic channel allocation, which we will henceforth abbreviate as *DDCA*. The focus of early work on DDCA [8, 9, 6] was on maximizing channel reuse in various cells at the possible expense of interference among cells caused by more than one cell (within the minimum reuse distance of each other) deciding to use the same channel. Thus, for example, I and Chao [8] propose a scheme in which each cell maintains in the *augmented channel occupancy* matrix, information about which channels are currently in use by the cell in question, as well as by neighboring cells. When a new call needs to be serviced by the current cell, it identifies a free channel by consulting the occupancy matrix and assigns it to that call. The problem is that a neighboring cell might also, at about the same time, have decided, by consulting its occupancy matrix, to allocate the same channel to service one of its calls. The cells in this scheme do, of course, exchange messages with each other providing information about which channels they are currently using so that each cell can update its matrix to reflect not only its own channel usage but also that of neighboring cells; but the issues, well known in the distributed computing literature, that can arise as a result of such messages being sent simultaneously by the two processes to each other, or the fact that information that any cell may have about the other cells is necessarily slightly out-of-date, etc., are not addressed. In other words, the focus of the work, as we noted, is on maximizing channel use; the complementary problem of ensuring that neighboring cells do not make conflicting decisions that can lead to interference between the calls is given little attention[1].

Recently, researchers interested in DDCA have recognized the importance of preventing interference among cells. The focus of their work [2, 4, 20] is not only on maximizing channel use but also in ensuring that if two cells can interfere with each other, they do not simultaneously use the same channel.

The problem of ensuring that neighboring cells do not simultaneously use the same channel can be viewed as a problem of *resource sharing*. In effect, the channel in question is a resource and we want to ensure that the neighboring cells access the resource in a *mutually exclusive* manner. But there are some important differences between the DDCA problem and the classical mutual exclusion problem. First, and perhaps most important, is that while in standard mutual exclusion two processes are not allowed under any circumstances to use the resource at the same time, in our case the same channel can in fact be used simultaneously by several cells; what is not allowed is two (or more) cells within the minimum reuse distance of each other doing so. A second difference is that in DDCA we are dealing not with a single resource but a collection of them, each channel being a resource. (Note that here we compare only the "ensuring that neighboring cells do not simultaneously use the same channel" aspect of DDCA with the classical mutual exclusion problem. The "maximizing channel reuse" aspect of DDCA, of course, is major difference between DDCA and the classical mutual exclusion problem which does not try to optimize anything.) Despite these differences, the similarity between DDCA and mutual exclusion is so strong that we believe that the lessons learned from distributed mutual exclusion algorithms can help us design efficient distributed channel allocation algorithm.

One class of algorithms for solving the mutual exclusion problem is the token-based algorithms [3, 19, 28]. Here, there is a single *token* in the system; the process that currently holds the token is the one that is allowed to use the resource. A process that wants to have access to the resource has to wait until it is in possession of the token. Since there is only one token in the system, only one process can hold it at any moment, so mutual exclusion is guaranteed. This type of approach seems unsuitable to our situation because we do not, in fact, want to forbid two cells from using a given channel at the same time as long as these two

[1]In fact, in the scheme proposed in [8], if there are no free channels available to the current cell c_1, it might choose a channel that is currently in use by another cell d if d currently has a number of free channels available. To make this possible, the occupancy matrix of c_1 includes information about the number of free channels currently available to each neighboring cell including d. This information is also updated when the cells exchange information about each others' channel usage. But several other cells $c_2, c_3, \ldots$ might more or less simultaneously decide to use the same channel that c_1 chose and for the same reason as c_1, that d has a number of free channels available; following these simultaneous decisions by $c_1, c_2, c_3, \ldots$, the channel in question may be in use by a number of interfering cells. So, again the focus is on maximizing channel use with little attention being given to strict elimination of interference between the cells.

cells are sufficiently far apart.

The other class of algorithms for the standard mutual exclusion problem is non-token-based algorithms [3, 17, 18, 23]. While there are important differences between these algorithms, a common idea is that the various processes exchange information with each other about which process is currently using the resource, which processes are waiting to access the resource, etc. When a process wishes to access the resource, it sends appropriate messages to a subset of the processes and waits for appropriate responses from some processes in order to gain access; similarly, when a process is done with using a resource, it sends appropriate messages to a subset of processes, 'releasing' the resource. Sanders [24] introduced the idea of an *information structure* which consists of a number of sets (for each process in the system) in which appropriate information about the current state of the system, such as which process currently holds the resource, which others are waiting to access it, etc., is stored; [24] introduced a generalized mutual exclusion algorithm such that the other non-token-based algorithms can be considered as special cases, with the differences between them being represented in terms of the differences in the corresponding information structures. Sanders also established a necessary and sufficient condition that the information structure needs to satisfy in order for the generalized algorithm to implement mutual exclusion correctly.

In this paper, we first introduce the notion of *relaxed mutual exclusion (RME)* which represents precisely the kind of mutual exclusion that we want to achieve, i.e., that certain pairs of cells may not simultaneously use the same channel but certain other pairs may. Next we introduce a suitable notion of information structure that allows us to store information appropriate for the DDCA problem, and a relaxed-mutual-exclusion-algorithm that uses this information. We also establish a necessary and sufficient condition that the information structure must satisfy in order to implement RME correctly.

Relaxed mutual exclusion is only part of the problem. After all, if all we cared about was ensuring that certain pairs of cells do not simultaneously use the same channel, we can easily achieve that by not allowing any cell to use any channel at any time! The analog, in the standard distributed system setting, would be that we want not only mutual exclusion but also freedom from deadlock. In the distributed system it is generally acceptable for one process to wait for quite a while from the time it requests access to the resource to the time it gets the access; in other words, freedom from deadlock simply means that we have a guarantee that if the process waits long enough it will ultimately get access to the resource. In our case this is not satisfactory because if a cell needs a channel in order to service a new call, it is generally not acceptable for it to have to wait for a long period of time to get the channel. It would be preferable in such a situation to *reject* the call (by sending a message to the cell phone to the effect that no channels are available to complete the call). In our work, we show how such a feature can be added to our RME algorithm. The basic idea is to introduce a *reject* message using which a given cell may reject the request from another cell for a free channel if in fact it is impossible to satisfy the request in a timely manner.

Next we turn to the question of dealing with multiple channels. We show how the information structure can be enriched to include suitable information about multiple channels, and how our algorithm can be extended to use the information to allocate channels to various cells to meet the needs of the traffic in each cell. In effect, our approach decomposes the DDCA problem into five subproblems, each raising an interesting design issue. The subproblems are:

(1) How to implement RME for a single resource?

(2) How to resolve deadlocks?

(3) How to deal with multiple channels?

(4) How to design efficient information structures?

(5) How to implement efficient channel selection strategies?

The first four subproblems deal with the "ensuring that neighboring cells do not simultaneously use the same channel" aspect of DDCA, whereas the last subproblem is concerned with "maximizing channel reuse."

The rest of the paper is organized as follows. Section 2 clarifies the problems addressed in this paper and their relationships to each other. Section 3 proposes a general relaxed mutual exclusion algorithm for a single resource (channel), which includes an information structure and an algorithm; the necessary and sufficient conditions for a correct information structure is given and its correctness is proved. Section 4 presents a simple strategy for resolving/avoiding deadlocks. Section 5 extends the proposed general relaxed mutual exclusion algorithm to deal with allocating multiple channels. Section 6 explores possible information structures that can be used by the proposed algorithms. Section 7 takes a closer look at channel selection strategies in distributed channel allocation. Finally, section 9 proposes an example distributed channel allocation scheme, which is then analyzed and compared with other distributed channel allocation schemes by simulation study. Section 10 concludes the paper.

2 Problems and Their Relationships

In this section, we will briefly consider the problem of mutual exclusion, the problem of relaxed mutual exclusion, and the problem of channel allocation. Our main purpose in this section is to discuss the relations between these problems, thereby laying the groundwork for developing related solutions which we do in the rest of the paper.

2.1 Mutual Exclusion

A distributed system consists of N processes which communicate asynchronously via message passing [14, 26]. The processes have a simple structure in which they alternate between computations inside a critical section and computations outside the critical section. The problem of *distributed mutual exclusion* (ME) is to design a protocol to coordinate entries into the critical section so that at any instant at most one process is in the critical section.

2.2 Channel Allocation

A cellular network is regarded as a regular grid of hexagonal cells [7, 13, 15]. An $n \times m$ cellular network has n rows and m columns of cells. (Figure 1 shows a 7 × 7 cellular network.) The cell at row i and column j is denoted as (i, j). The *distance* between two cells is defined as the Euclidean distance between the centers of the two cells. Thus, if the radius of the cell is normalized as $\sqrt{3}/3$ (i.e., the Euclidean distance between the centers of cell $(0, 0)$ and $(0, 1)$ is 1), then the distance between any two cells (i_1, j_1) and (i_2, j_2) is $\sqrt{(i_1 - i_2)^2 + (i_1 - i_2)(j_1 - j_2) + (j_1 - j_2)^2}$.

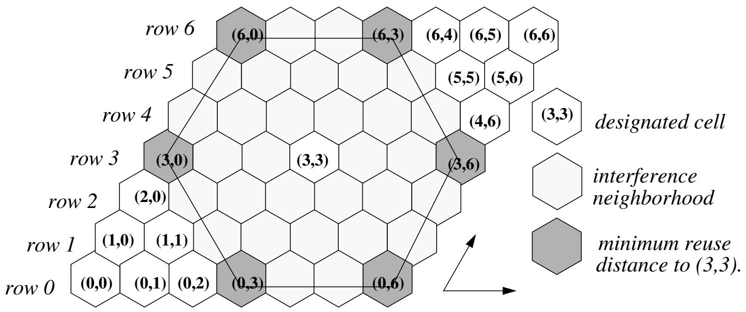


Figure 1: Hexagonal Cellular Model

A finite set of *channels* is available for the network. Each channel can be used to support a communication session[2]. A (real) number D_{min}, the *minimum reuse distance* is also specified. A channel can be used simultaneously by a number of different cells only if the distance between each pair of cells using the channel is greater than or equal to the minimum reuse distance. Thus, each cell c is associated with an *interference neighborhood* IN_c which is the set of cells whose distance to c is smaller than D_{min}:

$$IN_c = \{c' | dist(c, c') < D_{min}\}.$$

If a channel is being used by cell c, then it cannot be used by any cell in IN_c. Conversely, a channel is *available* for use by cell c if it is currently not being used by any cell in IN_c.

When a call arrives at a cell, the cell attempts to acquire a channel to serve the call. If there is no channel available to the cell, the call is *blocked*. Similarly, when the mobile user of an on-going call moves to another cell, the new cell attempts to acquire a channel to serve the handed off call. If no channel is available, the on-going call will be *dropped* (disconnected).

The *distributed dynamic channel allocation* (DDCA) problem is to design a protocol for the cells to exchange information so as to acquire channels without violating the interference constraint. The goals are not only to avoid interference but also to minimize the call blocking/dropping rate and to maximize wireless bandwidth utilization.

2.3 Relaxed Mutual Exclusion

A system consists of a set of *sites*, $S = \{s_1, s_2, \ldots, s_n\}$, and a set of *critical resources* $CR = \{r_1, r_2, \ldots, r_m\}$. At each site s_i there are k_i concurrent, independent processes, each of which repeatedly alternates between computations that need a critical resource and computations that do not need a critical resource. Among the sites there is a binary relation, $\otimes$, which is reflexive and symmetric, but not necessarily transitive. Two sites x and y are said to be *conflicting* or *interfering* with each other if and only if $x \otimes y$. Critical resources may be shared according to the following rules of *relaxed mutual exclusion*: 1) a given critical resource may be used simultaneously at different sites as long as no two of them are mutually interfering; and 2) at any

[2] In some systems, a channel can be used to support more than one communication session using time division multiplexing (TDM). In this case, a channel is called a *carrier*, and the channel allocation problem is referred to as the carrier allocation problem. All our results apply to such a system, with 'channel' being replaced by 'carrier'.

single site, a given critical resource may not be shared by two or more processes. The problem of *distributed relaxed mutual exclusion* (RME) is to design a protocol to dynamically assign critical resources to different sites so as to comply with the above rules.

2.4 Relationship among the Three Problems

It is easy to see that ME is a special case of RME, where there is only one critical resource, each site has only one process, and $x \otimes y$ for every pair of sites x and y. ME can also be viewed as a special case of CA, in which there is only one channel in the cellular system and the minimum reuse distance is *infinity*.

We can also draw a relation between CA and RME. Let S be the set of cells (or base stations) and define $x \otimes y$ if and only if the distance between site x and site y is less than the minimum reuse distance. Then the problem of allowing the same channels to be used simultaneously by different cells becomes a special case of RME. But we should note that there are also some important differences. In particular, in the CA problem, a cell requesting a channel does not really care *which* particular channel is allocated so long as it is able to use it without interference from other cells. By contrast, in the RME problem, when a particular site requests a critical resource, it is interested in acquiring a *specific* resource. Thus in the CA problem we have to concern ourselves with the question of choosing among numerous channels that may be available, when deciding which channel to allocate to a requesting cell. Indeed, as we noted in the Introduction, much of the work in the cellular literature has focussed on this question and a number of *channel selection* schemes have been proposed. In Section 7 we will consider the question of channel selection and see how some of these schemes can be integrated into our overall approach.

In the rest of the paper, we will use the terms relaxed mutual exclusion and distributed channel allocation, critical resources and channels, sites and cells interchangeably.

3 General Algorithm for Single Resource

As laid out in the introduction, our first step in attacking the RME problem is to develop a general algorithm that guarantees RME for a single resource. The algorithm we propose is a generalization of Sanders Algorithm [24] for mutual exclusion. We start by presenting the information structure we use; next we develop our algorithm; finally, we consider the necessary and sufficient condition that the information structure must satisfy in order to to ensure the correctness of the algorithm.

3.1 Information Structure and Variables

The concept of *information structure* forms the basis of Sanders Algorithm as well as our algorithm. The information structure of an ME (or RME) algorithm consists of two sets: a request set R_i and an inform set I_i.

- R_i: site i must acquire permission from all the sites in R_i before entering the critical session (or acquiring the critical resource).
- I_i: site i must inform all the sites in I_i when it leaves the critical session (or releases the critical resource).

In addition to R_i and I_i, our algorithm will also use the following three sets, which, however, are *not* part of the information structure:

- S_i: the *status set* for site i, is defined as $j \in S_i$ if and only if $i \in I_j$. This set is defined only for convenience.
- L_i: the set of requests that have been received by site i.
- CRU_i: according to i's knowledge, the sites in CRU_i are possibly using the critical resource at this moment.

3.2 Algorithm

Sites communicate with each other by message passing. The communications between any two sites are assumed to be First-In-First-Out (FIFO) and communication delay is arbitrary but finite. We use Lamport's logical clock scheme [12] to assign timestamps to events so that events at different sites can be totally ordered. Each message is assigned a timestamp. To break ties among requests, timestamp-based priorities are used. A request has a higher priority if and only if it has a smaller timestamp. Three types of messages are used: Request, Grant, and Release. The algorithm is described as a set of rules. The computations involved in each rule is considered as an atomic action.

The algorithm is a generalization of the Sanders algorithm. The generalization lies in Rule E, which specifies under what condition a site can grant a request. The generalization is not trivial because the condition cannot be too loose (or the algorithm will not guarantee RME), and it cannot be too tight (or the algorithm will not be general enough.) The condition must be just right so that Theorem 1 (presented later) is true. The algorithm described in this section is only concerned with RME. Deadlocks and other issues will be considered later. The following algorithm is called general relaxed mutual exclusion for a single resource (GRME).

ALGORITHM GRME

A. Requesting the critical resource.
When a site i wishes to acquire the critical resource, it sends a Request message to every site in its request set R_i; also, i sets its pending flag to indicate that it has a pending request. (The flag is cleared in step B.)

B. Acquiring the critical resource.
Site i acquires the critical resource only after it has received a Grant message from every site in its request set, R_i. On acquiring the critical resource, the site clears the pending flag. (The site cannot make another request until after the pending flag has been cleared.)

C. Releasing the critical resource.
When site i exits from the critical resource, it sends a Release message to every site in its inform set.

D. On receipt of a Request message.
On receipt of a Request message, a site i places the request in a request list L_i. (L_i is not necessarily a queue.)

E. Granting a request.

Site i may grant site j's request (stored in L_i) if both of the following two conditions are satisfied:

(a) either site i has no interference with j or it has no pending request of higher priority (or both);

(b) no site in CRU_i has interference with j.

When granting j's request, site i performs the following:

(a) send a Grant message to site j;

(b) remove j's request from L_i;

(c) if $j \in S_i$, then add j to CRU_i.

F. On receipt of a Release message.

When site i receives a Release message from site j, it removes j from CRU_i.

3.3 Necessary and Sufficient Conditions

Arbitrary choices of request, information, and check sets in the above algorithm do not necessarily result in a correct algorithm. In this section, we prove a necessary and sufficient condition for a correct information structure. (A particular information structure is correct if the resulting GRME algorithm implements relaxed mutual exclusion.)

Theorem 1 *Assume $I_i \subseteq R_i$ for every site i. A necessary and sufficient condition for Algorithm GRME to guarantee relaxed mutual exclusion is that for any two conflicting sites i and j their information structures satisfy $[(i \in I_i \cap R_j) \wedge (j \in I_j \cap R_i)] \vee (I_i \cap I_j \neq \emptyset)$.*

Proof. *Sufficiency.* Suppose the condition of the theorem is satisfied. Let i and j be any two mutually conflicting sites. We show that the two sites will never use the critical resource at the same time. There are two cases:

Case 1: $(i \in I_i \cap R_j) \wedge (j \in I_j \cap R_i)$. Two scenarios need to be considered. In the first scenario, both i and j send their requests simultaneously. Assume that i's request has a higher priority than j's. Since site i has a pending request of higher priority, it will not send a Grant until it has acquired and released the critical resource. The second scenario occurs when a site, say j, decides to acquire the critical resource after it has received and granted a Request from site i. When i receives j's request, either it still has its own request pending (which has a lower timestamp, thus higher priority, than j's request) or it has already acquired the critical resource (in which case $i \in CRU_i$ as $i \in S_i$). Site i cannot grant j's request until after it has released the critical resource.

Case 2: $\neg[(i \in I_i \cap R_j) \wedge (j \in I_j \cap R_i)]$. In this case, there must be a $k \in I_i \cap I_j$. For this k, $\{i, j\} \subseteq S_k$. Once k grants a request to i (or j), i (or j) will be in CRU_k. This will prevent k from granting the other site's request.

In both of the above cases, i and j will never occupy the critical resource at the same time. This establishes the sufficiency of the condition.

Necessity. Assume that the condition stated in the theorem is false. It suffices to construct a scenario in which relaxed mutual exclusion is violated. Since the condition is false, there exist at least two conflicting sites i, j such that $i \notin I_i \cap R_j$ or $j \notin I_j \cap R_i$ and $I_i \cap I_j = \emptyset$. Without loss of generality, assume $i \notin I_i \cap R_j$.

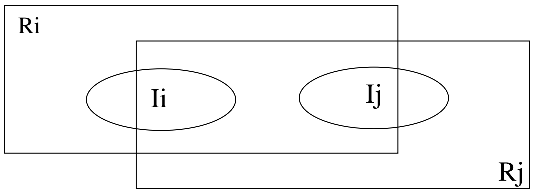


Figure 2: Information Structure

Let i and j each have a request of priority(i) and priority(j), respectively. We first show that if the requests arrive at a site in a certain order, they will both be granted. We then create a scenario in which i's and j's requests arrive at each site in that certain order. We consider three classes of sites: i, j, and other sites k.

First, consider site i. We have assumed $i \notin I_i \cap R_j$, so $i \notin I_i$ or $i \notin R_j$. The case $i \notin R_j$ is trivial since j does not ask i for permission. If $i \notin I_i$, then i will never be in CRU_i; it may grant its own request and then grant j's request (either immediately or after acquiring the critical resource, depending on whether $\text{prioity}(i) < \text{priority}(j)$ or the other way around). Thus, in our scenario, we will let i's request arrive at i sooner than j's request.

Now, consider site j. There are two cases depending on whether j is or is not in I_i.

- Case 1: $j \notin I_i$. In this case, $i \notin S_j$; thus, j keeps no record of granting i's request. It may grant i's request and then grant its own.
- Case 2: $j \in I_i$. In this case, since $I_i \cap I_j = \emptyset$, $j \notin I_j$ (and thus $j \notin S_j$) and so j may grant its own request and then grant i's request (either immediately or after acquiring the critical resource, depending on whether $\text{prioity}(i) > \text{priority}(j)$ or the other way around).

Now, consider those sites $k \neq i, j$. Obviously, if $k \notin R_i \cap R_j$ or $k \notin I_i \cup I_j$, then k is not capable of preventing i and j from violating relaxed mutual exclusion. Thus, we concentrate on those sites in

$$R = [R_i \cap R_j \cap (I_i \cup I_j)] - \{i, j\} = [(R_i \cap I_j) \cup (R_j \cap I_i)] - \{i, j\}.$$

Split R into two sets, A and B, as illustrated in Figure 2:

$$A = (R_i \cap I_j) - \{i, j\}$$

$$B = (R_j \cap I_i) - \{i, j\}$$

It is obvious that $A \cap B = \emptyset$ and $A \cup B = R$. Furthermore, since $I_i \cap I_j = \emptyset$, $i \notin S_k$ for all $k \in A$ and $j \notin S_k$ for all $k \in B$. That is, for every site $k \in A$ ($k \in B$), k does not keep record of granting i's (j's) request. Thus, if i's request arrives at each site in A before j's request, and j's request arrives at each site in B before i's request, then both i's and j's requests will be granted by all sites in R.

In summary, to make i and j violate relaxed mutual exclusion, we will let their requests arrive and be granted at various sites as follows (where $(i, \_, \_, j)$, for example, means that i's and j's requests are garnted at t_1 and t_4, respectively):

	priority(i) $>$ priority(j)		priority(i) $<$ priority(j)	
	$j \notin I_i$	$j \in I_i$	$j \notin I_i$	$j \in I_i$
Site i	$(i, \_, \_, j)$	$(i, \_, \_, j)$	$(i, j, \_, \_)$	$(i, j, \_, \_)$
Site j	$(i, j, \_, \_)$	$(\_, j, i, \_)$	$(i, j, \_, \_)$	$(\_, j, i, \_)$
Sites in A	$(i, j, \_, \_)$	$(i, j, \_, \_)$	$(i, j, \_, \_)$	$(i, j, \_, \_)$
Sites in B	$(\_, j, i, \_)$	$(\_, j, i, \_)$	$(\_, j, i, \_)$	$(\_, j, i, \_)$

Based on the above discussion, we now create a scenario to fool all the sites in $R \cup \{i, j\}$. (For clarity, we assume without loss of generality that $R_i, R_j \subseteq R \cup \{i, j\}$; the sites outside $R \cup \{i, j\}$ are not capable of preventing i and j from violating mutual exclusion.) Let $t_0 < t_1 < \cdots < t_4$.

(0) Let the system start at time t_0, with $CRU_k = \emptyset$ for every site k in $R \cup \{i, j\}$.

(1) At time t_1, site i issues a request, which is immediately received and granted by i if $i \in R_i$; by j if $j \in R_i - I_i$; and by every site in A.

(2) At time t_2, site j generates a request, which is received and immediately granted by i if $i \in R_j$ and priority(i) $<$ priority(j); and by every other site in R_j.

(3) At time t_3, i's request is received and granted by j if $j \in R_j$ and $j \in I_i$; and by every site in B.

(4) Shortly after time t_3, i acquires the critical resource and clears its pending flag.

(5) At time t_4, j's request is received and granted by i if $i \in R_j$ and priority(i) $>$ priority(j).

(6) Shortly after time t_4 j acquires the critical resource.

Thus, both sites use the critical resource simultaneously, in violation of relaxed mutual exclusion. This establishes the necessity of the condition stated in the theorem. ■

3.4 Application to Mutual Exclusion

Sanders' algorithm is a special case of the General Relaxed Mutual Exclusion (GRME) algorithm. When every two sites are mutually conflicting (as in the case of mutual exclusion), the variable CRU_i may contain at most one element. Item E of the GRME algorithm can be rewritten as:

E. Granting a request.
A site i may grant a request (say from site j) in L_i if $CRU_i = \emptyset$ and site i has no pending request of higher priority. When granting j's request, site i performs the following:

(a) send a Grant message to site j;
(b) remove j's request from L_i;
(c) if $j \in S_i$ then let $CRU_i = \{j\}$.

With rule E written as above, this becomes essentially equivalent to the algorithm of [24]. The main difference is that [24] uses a priority queue, whereas our L_i need not be a priority queue. Priority queues help avoid deadlocks but they are not necessary for achieving mutual exclusion.

4 Resolving/Avoiding Deadlocks

A deadlock occurs if there exists a cycle of sites $(s_0, t_0, s_1, t_1, \ldots, s_n, t_n)$ such that (a) for each i, s_i has received a Grant message from t_i and is waiting for a Grant from t_{i+1}, and (b) for each i, t_i is unable to grant the request from s_{i-1} because s_{i-1} has conflict with s_i. In the literature, many deadlock resolution/avoidance algorithms have been proposed. However, most of these algorithms have a long response time for mutually exclusive resource acquisition and may not be appropriate for real-time distributed systems such as a mobile cellular network.

One effective way to avoid deadlocks is by use of Reject messages. After a site receives a request, it either grants or rejects the request within a finite amount of time. Thus, the algorithm is modified as follows:

ALGORITHM GRME_with_Grant/Reject

A-F. Same as in GRME.

G. Rejecting a request.
If a site i is unable to grant a request in L_i within a pre-specified amount of time, it rejects the request by sending a Reject message to the requesting site and, at the same time, removing the request from L_i.

H. On receipt of a Reject message.
When a site i receives a Reject message, it aborts its request (if it has not been aborted) by clearing the pending flag and sending a Revoke message to every site in I_i.

I. On receipt of a Revoke message.
When a site i receives a Revoke message from j, it performs the following:

(a) If j's request is in L_i, remove the request.
(b) If $j \in CRU_i$, then remove j from CRU_i.

J. After aborting a request
If site i aborts a request, it cannot generate another request until after it has received a response (grant or reject) from every site in R_i.

Correctness. Since any site requesting for the critical resource will receive a response (either a grant or a reject) from every site in its request set, no deadlock may occur. Also, with Rule J, no grant to a site's request will be mistaken as a grant to the site's another request. Theorem 1 still holds for the modified GRME algorithm. The "sufficiency" part of the proof needs only minor modification. The "necessity" part needs no change at all.

Performance. The advantage of the Grant/Reject scheme is the short response time — the requesting site knows quickly whether it can or cannot use the critical resource. This property is particularly useful in real-time application such as distributed channel allocation. (Please see Section 5.1 for related discussion.) If response time is not a critical concern, there are many deadlock detection/resolution algorithms in the literature that can be incorporated into or executed in conjunction with the GRME algorithm to resolve possible deadlocks.

The disadvantage of this scheme is that it does not guarantee progress property: it is possible that no site can obtain the critical resource even though it is free. For example, if a set of sites submitted requests for a resource simultaneously and forms a deadlock, the use of the Grant/Reject scheme resolves the deadlock, but as a side-effect, all the requests are rejected and none site can acquire the resource even if it is available. In the worst case, if all the rejected sites start to send request again right after the rejections, the same non-progress situation may repeat. One simple solution that can be borrowed from slotted-ALOHA protocol is to let each rejected site backs off a random length of time and then attempts another request.

An interesting problem for future work is to design a deadlock avoidance/resolution scheme with progress property.

5 Dealing with Multiple Channels

The algorithms described in Section 3 consider only one channel (considered as the critical resource). Now we discuss how to deal with multiple channels, which is necessary for a distributed channel allocation. Let $CH = \{r_1, r_2, \ldots, r_n\}$ be the set of channels in a cellular system. In the following, we describe three approaches to dealing with multiple channels.

5.1 Sequential Search

Let Algorithm(r) be any algorithm for the single-channel distributed channel allocation problem, with r being the channel. The sequential search strategy works as follows:

1. When a cell wishes to acquire a channel, it selects a channel $r \in CH$.
2. Use Algorithm(r) to request for r.
3. If the request fails, try another channel r' and repeat the second step.

Two comments are in order. First, for this approach to work correctly, the single-channel algorithm must have the property that every request for a channel either succeeds or fails within a finite amount of time. Second, a cell may have different information structures for different channels. Thus, each time Algorithm(r) is executed, a different information structure, I_i and R_i, may be used. For distributed channel allocation, this may improve the system performance.

Design Issues. One major design issue in sequential search is how to select a channel in step 1. Clearly, the cell should choose a channel that is likely to be available. We will discuss in Section 8 possible schemes for collecting information so as to know which channels are available or likely to be available. Now, suppose a number of channels are found to be potentially available, which of them should be selected? We will address this question in Section 7.

5.2 Parallel Search

The basic idea behind parallel search is that when a cell wants to acquire one or more channels, it executes Algorithm(c) simultaneously for all $c \in CH$ to determine whether it can acquire c. Conceptually, n copies of the Algorithm() are executed in parallel, where n is the number of channels. But blindly executing n

independent copies of the algorithms is inefficient. One design issue here is how to cut the message overhead and how to execute n copies of the algorithm collectively, rather than independently. The following describes one possible design using GRME_with_Grant/Reject as the underlying algorithm. In essence, the idea is as follows: When a cell i responds to a request from cell j, it sends j a *set* of channels F_i, which contains all the channels that are, according to the information that i has, currently usable by j. Once j receives this information from all the sites it has sent its requests to (in other words, all the sites in R_j), it selects channels that are in *all* the F_i's it has received, and sends a message to each site in I_j informing them of the set of channels it has decided to acquire. Thus, rather than trying one channel after another in a sequential manner as in the case of *Sequential Search*, here we are considering all possible channels in parallel; but note that the information structure we use here is the same for all channels whereas, as we noted above, this need not be the case for the Sequential Search algorithm.

In Algorithm GRME, each site i maintains a CRU_i. Here, since there are multiple channels, cell i will maintain a separate CRU_i for each channel. The CRU_i for channel c is denoted as $CRU_i(c)$.

ALGORITHM Parallel GRME_with_Grant/Reject

A. Requesting channels.

When a cell i wishes to acquire one or more channels, it sends a Request message to every cell in its request set R_i; also, i sets its pending flag to indicate that it has a pending request. (The cell cannot make another request until after the pending flag has been cleared in step B or I.)

B. Acquiring channels.

After cell i has received a Grant(F_j) message from every cell j in R_i, it performs the following:

(a) Compute $F = \cap\{F_j : j \in R_i\}$.

(b) Select a set, F', of channels in F according to some selection policy.

(c) Send an Acquire(F') message to every cell in I_i.

(d) Clear the pending flag.

Note that if $F = \emptyset$, then the search fails; in this case, $F' = \emptyset$ in the Acquire(F', r) message.

C. Releasing a channel.

When cell i wants to release a channel c, it sends a Release(c) message to every cell in its inform set I_i.

D. On receipt of a Request message.

On receipt of a Request(r) message, cell i places the request in a request list L_i.

E. Responding to a request.

Cell i may grant j's request in L_i if both of the following two conditions are satisfied:

(a) Either cell i has no interference with j or it has no pending request of higher priority (or both);

(b) $F_i \neq \emptyset$, where F_i is the set of all channels x such that none of the cells in $CRU_i(x)$ has interference with cell j.

When granting j's request, cell i performs the following:

(a) Send a Grant(F_i) message to cell j, where F_i is as computed above, and remember this F_i as F_{ij} for use in steps G and J.

(b) Remove j's request from L_i;

(c) If $j \in S_i$ then add j to $CRU_i(x)$ for each x in F_i.

F. On receipt of a Release message.

When cell i receives a Release(c) message from cell j, it removes j from $CRU_i(c)$.

G. On receipt of an Acquire message.

When cell i receives an Acquire(F') message from cell j, it removes j from $CRU_i(x)$ for every $x \in F_{ij} - F'$.

H. Rejecting a request.

If cell i is unable to grant a request in L_i within a pre-specified amount of time, it rejects the request by sending a Reject message to the requesting cell and, at the same time, removes the request from L_i.

I. On receipt of a Reject message.

When cell i receives a Reject message, it aborts its request (if it has not done so) by clearing the pending flag and sending a Revoke message to every cell in I_i.

J. On receipt of a Revoke message.

When cell i receives a Revoke message from j, it performs the following:

(a) If j's request is still in L_i, then remove the request from L_i.

(b) Otherwise, remove j from $CRU_i(x)$ for every x in F_{ij}.

Correctness of parallel search: As mentioned in the beginning of this subsection, the parallel search algorithm is a "group" execution of n copies of Algorithm GRME_with_Grant/Reject. The algorithm will work correctly if it works correctly for each individual copy, i.e., if the algorithm works correctly for each individual channel. Focusing on a particular channel z and examining how the algorithm works for z, we see that as far as z is concerned, the Parallel GRME_with_Grant/Reject is essentially the same as GRME_with_Grant/Reject. For instance, consider step E. If $z \in F_i$, then as far as z is concerned, this step is the same as the counterpart (step E) of GRME_with_Grant/Reject. If $z \notin F_i$, then when the requesting cell, say j, receives i's Grant(F_i), it will find $z \notin F$ (in step B) and thus not acquire z. The net effect of step E for z is equivalent to sending a Reject(z); and the net effect of step B is equivalent to receiving a Reject(z). For channel z, since $z \notin F'$, the Acquire(F') sent by j has the effect of Revoke(z); both messages will cause the receiving cell to remove j from $CRU_i(z)$.

Theorem 1 still holds for Parallel GRME_with_Grant/Reject.

5.3 Hybrid Search

Distributed channel allocation may impose very stringent requirements on channel acquisition delay. For example, a call in handoff needs a new channel in the new cell within a very short time period. If the new channel is not acquired in time, the call is dropped. A cell may also request two or more channels at a time. One such example is the future third generation multimedia mobile network, in which a call may need multiple channels. As described in the preceding section, parallel search allows a cell to acquire more than one channel at a time and would suit such multimedia applications. A disadvantage of parallel search is that when a cell receives a request, it may grant unnecessarily too many channels (the entire set F_i in step E of Parallel GRME).

Hybrid search is a combination of sequential and parallel search. when a cell wishes to acquire m channels, it actually requests for l channels ($l \geq m$). The proper value of l is chosen such that after one round of search, m or more than m channels can be acquired. On the other hand, the value of l should be as small as possible so as to reduce the probability of collision among conflicting cells. When l is chosen appropriately, both the acquisition delay and the message complexity can be kept low. Exchanging resource usage information between cells is an option that can be incorporated into the algorithm to further improve the performance of the algorithm. Due to space limit, the details of hybrid search are not presented here.

6 Information Structure

Theorem 1 describes a necessary and sufficient condition for a correct information structure. For the problem of distributed channel allocation, the design of information structure is very important; it may significantly affect the system's performance in terms of message complexity, channel acquisition delay, as well as bandwidth efficiency of a distributed channel allocation scheme. In this section, we describe some commonly used information structures and propose some new ones.

As noted earlier, a cellular network is a regular grid of hexagonal cells, with the cell at row i and column j denoted as (i, j); Figure 1 shows a 7×7 cellular network. In practice, the minimum reuse distance is chosen to be the distance between the centers of two cells, say, $(0, 0)$ and (a, b). This distance is known to be $D_{min} = \sqrt{a^2 + ab + b^2}$ if the distance between $(0, 0)$ and $(0, 1)$ is normalized as 1 [15]. The nearest "co-channel" cells of any cell c are those cells whose distance to i is exactly D_{min}; these cells, to maximize channel reuse, should use the same set of channels as cell c – thus the name co-channel cells. The nearest "co-channel" cells of any cell c can be located as follows [15]: Starting at cell c, move a cells along any chain of hexagons; turn counter-clockwise 60 degrees; move b cells along the chain that lies on this new heading. Every cell has six nearest co-channel cells, except for those on or close to the boundary of the cellular network. In Fig. 1, $a = 3$ and $b = 0$; the six nearest co-channel cells of $(3, 3)$ are $(0, 3)$, $(0, 6)$, $(3, 6)$, $(6, 3)$, $(6, 0)$, $(3, 0)$. In the right figure of Fig. 4, $a = 3$ and $b = 2$; the nearest co-channel cells of the center cell are marked with an "A".

Given two cells x and y, x is said to be a *co-channel cell* of y if there is a sequence of cells ($c_1 = x, c_2, \ldots, c_{k-1}, c_k = y$) such that c_i is a nearest co-channel cell of c_{i+1}, $1 \leq i \leq i - 1$.

Given a minimum reuse distance D_{min}, the set of all cells in the system can be partitioned into N subsets $G_1, G_2, \ldots, G_N$ such that all members in a G_i are co-channel cells of each other, where $N = D^2$ [15]. In Figure 3, $D = 3$ and thus there are $N = 9$ subsets of co-channel cells; all cells in a same subset are labeled with a same number. We will refer to each G_i as a *co-channel set*.

Also recall that each cell c is associated with an *interference neighborhood* IN_c where

$$IN_c = \{c' | dist(c, c') < D_{min}\}.$$

Given the minimum reuse distance $D_{min} = \sqrt{a^2 + ab + b^2}$, the size of IN_c can be approximated as follows. Let $d = \lfloor D_{min} \rfloor$. (Note that $d = a + b$ if $ab = 0$; and $d = a + b - 1$, otherwise.) When $ab = 0$, the number

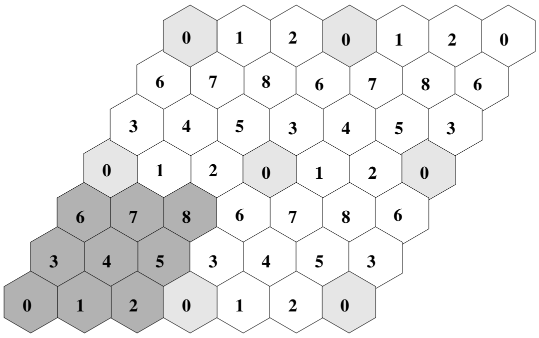


Figure 3: With $D_{min} = 3$, $N = 9$.

of cells in IN_c is $|IN| = 1 + \sum_{i=1}^{d} 6i - 6 = 3d(d+1) - 5$. When $ab \neq 0$, $|IN| = 1 + \sum_{i=1}^{d} 6i = 3d(d+1) + 1$. In any case, $|IN| \approx 3d^2$.

Using the terminology of relaxed mutual exclusion, $IN(i)$ is the set of all sites which have interference with site i. Recall that the information structure for a site i consists of sets, R_i and I_i, and they must satisfy the condition of Theorem 1. The following are some possible information structures.

• Centralized structure: $I_i = R_i = \{c\}$, where c is a particular site responsible for all channel allocation in the system. (In practice, site c is typically the *mobile switching center* to which all base stations in a region are connected [7].) This structure satisfies the condition of Theorem 1 as $I_i \cap I_j \neq \emptyset$ for any i and j.

• Partially centralized structure [2, 4]: Let Z be any arbitrary but fixed co-channel set of cells. (For instance, in Figure 3, Z can be the set of all cells marked "0".) The algorithms in [2, 4] have the following information structure. For $i \in Z$, $I_i = R_i = \{i\}$; and for $i \notin Z$, $I_i = R_i = IN_i \cap Z$. For example, in Figure 3, suppose we let Z be the set of all 0-cells (i.e., all cells marked with 0). Then for any 1-cell i, R_i consists of the four 0-cells which "enclose" cell i. For any 4-cell i, R_i consists of the three 0-cells that "enclose" cell i. It is not hard to see that regardless of D_{min}, $|Z \cap IN_i| = 3$ or 4. Note that this structure is "unsymmetric"; it favors the cells in Z. This structure satisfies the condition of Theorem 1. To see this, observe that if i and j are in each other's *IN*, then there is a cell k such that i and j are both in IN_k, from which it follows that $k \in IN_i \cap IN_j$ and thus $k \in I_i \cap I_j \neq \emptyset$.

• Trivial structure [4, 20]: $I_i = \{i\}$ and $R_i = IN_i$.This information structure, though simple, is not efficient. Since all the interfering cells are in the information structure, the message complexity, acquisition delay as well as deadlock probability are higher. This structure satisfies the condition of Theorem 1: $i \in I_i \cap R_j = \{i\}$ and $j \in I_j \cap R_i = \{j\}$ for any two interfering cells i and j.

• Mercedes-logo structure: Figure 4 illustrates the structure. Let i be the center cell labeled with A, whose nearest co-channel cells are also labeled with A. The light-shaded area indicates the *IN* of i, denoted as IN_i. The dark-shaded area is the information structure, i.e., $R_i = I_i$ = "the Mercedes logo." As for S_i, recall that $j \in S_i$ iff $i \in I_j$; geometrically, S_i is symmetric to I_i with respect to cell i.) This structure satisfies the condition of Theorem 1; in fact, if i and j are in each other's *IN*, then $I_i \cap I_j \neq \emptyset$. Using the defined variable d, $|R_i| = |I_i| = |C_i| = 3d - 2$ (if $a = 0$ or $b = 0$) or $3d + 1$ (if $a \neq 0$ and $b \neq 0$).

• Ring structure: Figure 5 illustrates the ring structure. The light-shaded area is the IN of the center

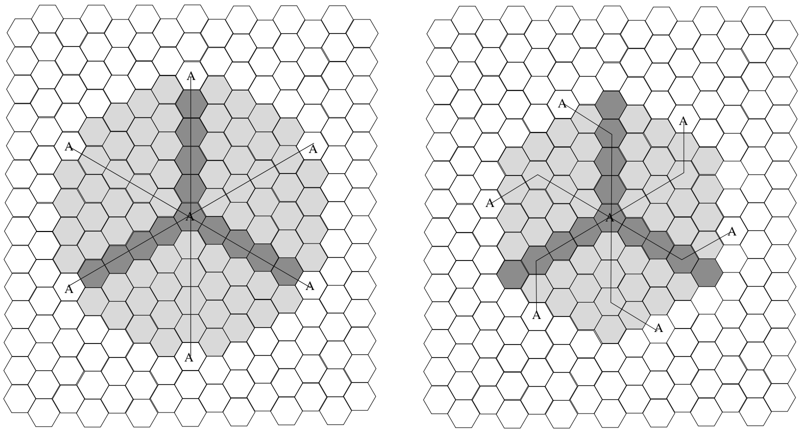


Figure 4: The Mercedes-logo structure. Left: $a = 5$ and $b = 0$. Right: $a = 3$ and $b = 2$.

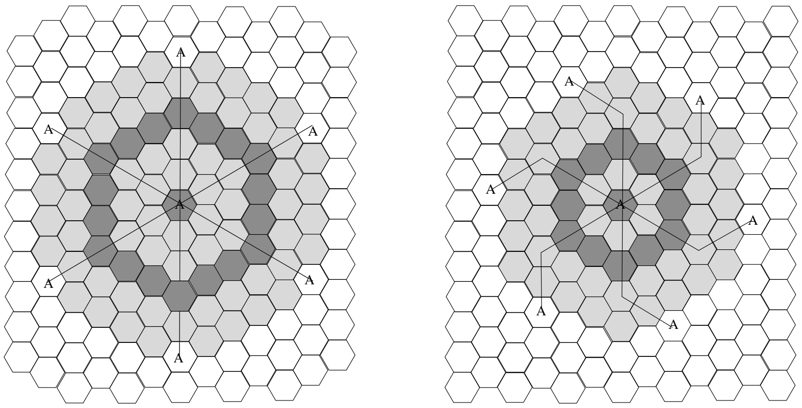


Figure 5: The ring structure. Left: $a = 5$ and $b = 0$. Right: $a = 3$ and $b = 2$.

cell i labeled by A. The dark-shaded area is the information structure: $R_i = I_i$ = "the ring." In general, the ring structure of cell i is the ring of cells which are $\lceil d/2 \rceil$ cells away from i. (In Fig. 5, $d = a + b = 5$ for the left structure and $d = a + b - 1 = 4$ for the right one..) The size of this information structure is $6\lceil d/2 \rceil$, approximately equal to that of the Mercedes-logo structure. It is obvious that for any two interfering cells i and j, I_i and I_j have a nonempty intersection. The condition of Theorem 1 is satisfied.

All of these information structures satisfy the conditions of Theorem 1 and are thus "correct." Besides correctness, we also need to address the question of efficiency. One measure of efficiency is the *size* of the information structure. A smaller information structure would reduce message complexity as well as deadlock probability since a smaller information structure implies that fewer processes are involved in making decisions about which channel will be allocated to which cells. At the same time, it may reduce fault tolerance since failure of some of these processes would affect many cells. Among the above mentioned structures, the centralized structure is smallest, and the trivial structure the largest.

Size is not the only concern in choosing an information structure. Maekawa [17] argues that for truly distributed algorithms, the following two *symmetry properties* are required or desirable: 1) that all request sets R_i's have the same size and 2) that every cell be in the same number of R_i's. The Mercedes-logo, ring, and trivial structures all have these properties, but not the centralized and partially centralized structures.

When IN_i is the set of all sites in the system (which is the case for the classical mutual exclusion problem), Maekawa [17] considers the question of the minimum possible size of the information structure with the above two symmetry properties. He shows that the minimum possible size in this case is approximately $\sqrt{N}$, where N is the number of sites in the system. It is not clear whether Maekawa's result — that the minimum possible size of information structure with the two symmetric properties is approximately $\sqrt{|IN_i|}$ — will hold when $|IN_i| \neq N$. Nevertheless, we feel that Maekawa's result is still a useful benchmark.

For cellular networks, $|IN_i| \approx 3d^2$, where $d = \lfloor D_{min} \rfloor$. If Maekawa's formula still applies, the minimum possible size of information structure (with symmetry properties) would be $\sqrt{3}d$. Of the three information structures with the symmetry properties, both the Mercedes-logo and the ring structure have a size close to this value.

7 Selection Strategies

In step 1 of the sequential search algorithm (Section 5.1), the cell selects a channel r according to some selection strategy. Again in step B-a of the parallel search algorithm (Section 5.2), the cell acquires a channel in a set according to some selection policy. Channel selection strategy is an important issue in both centralized and distributed channel allocation. Basically, the question is, given a set of available channels, which channel should be acquired? Previous studies [11] indicate that efficient channel selection strategies can result in near-optimal reuse of channels among cells.

Most previously proposed channel selection strategies were studied for the centralized system [11]. They assume complete and updated knowledge of the channel usage information in a cell's neighborhood. This may not hold in the context of distributed systems, even if each cell broadcast channel acquisition/release messages to its neighbors immediately after each event. One reason is the delay in broadcasting channel acquisition/release information. Another is that several cells may request/acquire/release channels at approximately the same time. Moreover, having each cell broadcast channel usage information to all its neighboring cells is not desirable since this will likely result in high message complexity.

All this does not necessarily invalidate the importance of channel selection strategies; rather, it is an indication of the complexity of channel selection in distributed channel allocation. There are two aspects to this complexity. First, efficient protocols to exchange channel usage information have to be designed. One possible approach is to collect such information during the process of requesting channels; we will adopt this approach in one of the schemes in the next section. Second, we have to address the question of which channel to acquire if more than one is available?

The focus of most of the dynamic channel allocation schemes in the literature [11] is primarily on efficient channel selection strategies. The question of efficient protocols for exchanging channel usage information has received much less attention; as noted earlier, to an extent the work presented so far in this paper tries to address this imbalance. But now let us move on to the channel selection aspect. Most commonly, the selection strategy is priority-based, i.e. each channel is associated with an acquisition priority in each cell. When a channel is needed, the potentially available channel with the highest acquisition priority is selected. The problem of channel selection then becomes one of assigning priorities to channels; the aim being, to assign priorities to channels in ways that maximize channel reuse.

As an example, consider the selection strategy from [5]. This scheme is a two-step dynamic-priority (TSDP) channel selection strategy. For any given cell i, channels are classified as *primary* or *secondary* based on the degree of reuse of each channel in i's nearest co-channel cells. The primary channels are those that are used by at least λ nearest co-channel cells, where λ ($1 \leq \lambda \leq 7$) is a constant. Primary channels have higher acquisition priority than secondary channels. The acquisition priority of a primary channel is higher if it is used by more co-channel cells. The acquisition priority of a secondary channel is higher if its cost is lower, i.e. the use of the channel by i will make the channel unavailable to fewer "new" cells ("new" in the sense that the channel is currently available, but will become unavailable if i uses it). The motivation behind this scheme is to maximize the use of primary channels among co-channel cells while reusing secondary channels in a compact way. The primary disadvantage of this scheme is the high overhead in calculating channel priorities; it requires channel usage information of *all* cells in i's *outskirts interference neighborhood*, $OIN_i = \cup\{IN_x : x \in IN_i\}$. In the next section, we will use the TSDP selection scheme in conjunction with some of the earlier information structures that we discussed in Section 6.

8 Designing Distributed Channel Allocation Schemes

Having explored many of the issues involved in designing distributed channel allocation schemes, in this section we use these results to guide us in the design of distributed channel allocation schemes. Our goal here is to demonstrate how to integrate channel selection schemes (including channel usage information collection) with channel acquisition algorithms, consider how the information structure can affect the performance of a scheme, and discuss ways to use different search algorithms to tackle different requirements of calls (such as handoff calls). We will also compare the performance of the proposed schemes with other well-known schemes by simulation studies.

8.1 Algorithms

8.1.1 TSDP with Mercedes-logo Information Structure

Basis of the algorithm: We use GRME_with_Grant/Reject (which was presented in Section 4) as the basis of the algorithm.

Information structure: The first thing to do in applying GRME_with_Grant/Reject is to define an information structure. Suppose that we use the Mercedes-logo information structure. For ease of presentation, let us assume $a = 2$ and $b = 1$ in the formula $D_{min} = \sqrt{a^2 + ab + b^2}$. (This value of D_{min} is used in many cell-phone networks.) The Mercedes-logo information structure thus consists of 7 cells as shown in Figure 6.

Sequential/Parallel search: For this demonstration, we choose the sequential search. As pointed out in Section 5.1, we need a scheme to decide which channels are likely to be available; and if there is more than one such channel, we need a policy for selecting one of them. The former requires a scheme to collect channel usage information; the latter, a channel selection strategy.

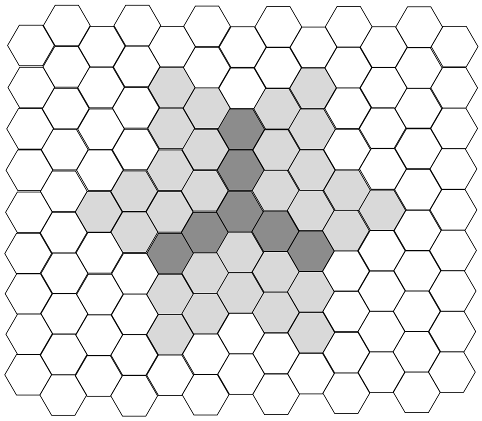


Figure 6: Channel Usage Information Coverage

Collecting channel usage information: Cell i wants to know which channels are available for its use. A simple way is to ask the cells in IN_i which channels they are currently using [2]. This will require $2|IN_i|$ messages. Here we propose a more message-efficient method. Recall that in algorithm GRME_with_Grant/Reject, each cell j maintains for each channel r a set $CRU_j(r) \subseteq S_j$ that represents j's knowledge of who are possibly using channel r. For the Mercedes-logo information structure, the set $\cup\{S_j : j \in I_i = R_i\}$ (the gray-shaded area in Figure 6) contains IN_i. Thus, if every cell $j \in R_i$ sends its $CRU_j(r)$ for every channel r to cell i, then i will know which channels are potentially available — channel r is potentially available for cell i if $\cup\{CRU_j : j \in R_i\}$ has no intersection with IN_i. Based on this idea, we propose the following scheme for collecting channel usage information:

- In executing GRME_with_Grant/Reject, whenever cell i has to send a message (such as Request, Grant, Reject, Release, etc.) to cell j (where $j \in I_i = R_i$), attach to the message the set $CRU_i(r)$ for every channel r. The attached sets are very small. Indeed, for the Mercedes-logo structure, each $CRU_i(r)$ contains no more than three elements.

Channel selection strategy: We will adopt the TSDP strategy as described in Section 7. The information needed for TSDP — namely, the channel usage information at the nearest co-channel cells — is available, since the channel information coverage (the gray area in Figure 6 includes the nearest co-channels).

The proposed scheme uses sequential search. If the request for a channel fails, the cell has to search for another channel. For handoff calls, it is most desirable to do one request and succeed. One solution is to use hybrid search, i.e. to let a cell request/search for several channels at one time. The number of channels to search using hybrid search is a design parameter and could vary under different situations.

8.1.2 TSDP with Trivial Information Structure

The Basic Update Scheme described in [4] is an instance of GRME_with_Grant/Reject using the sequential search and trivial information structure. It can work with any selection strategy. For performance studies in the next section, we will use the TSDP strategy.

8.1.3 Distributed Borrowing Scheme

In the distributed channel borrowing scheme [2, 11], each cell i is pre-allocated a number of channels. These channels are the cell's *primary* channels. The other channels (which are pre-allocated to other cells in IN_i) are *secondary* channels for i. When i wishes to acquire a primary channel, it does not send any request to other cells; instead, it consults only itself. But when i wishes to acquire a secondary channel, it first queries all cells in IN_i to find out which channels can be borrowed, select a channel, and send a borrowing-request to those cells (typically 3 or 4 such cells) which own the selected channel. This, algorithm can be regarded as an instance of GRME_with_Grant/Reject, where the sequential search and the partially centralized information structure are used, and where a cell sends a query to every cell in IN_i to collect channel usage information.

8.2 Analysis and Simulation Results

There are many criteria to evaluate the performance of a channel allocation scheme. Some of the important ones are call blocking/dropping rate, bandwidth utilization, message complexity, and channel acquisition delay. Call blocking rate and call dropping rate together constitute the call failure rate. Bandwidth utilization is defined as the percentage of system bandwidth capacity used for transmitting communication good-put (or useful user packets). Message complexity refers to the number of messages exchanged for each channel acquisition/release while channel acquisition delay is to measure the average time for a cell to acquire a channel. In this section, we will provide performance evaluation of the three schemes described in preceding sections.

Let d be as defined in Section 6. Recall that $|IN_i| \approx 3d^2$, and that the Mercedes-logo information structure consists of $3d$ (plus 1 or minus 2) cells. Assume the message transmission delay between any two cells to be T. The message complexity and channel acquisition delay can be analyzed as follows.

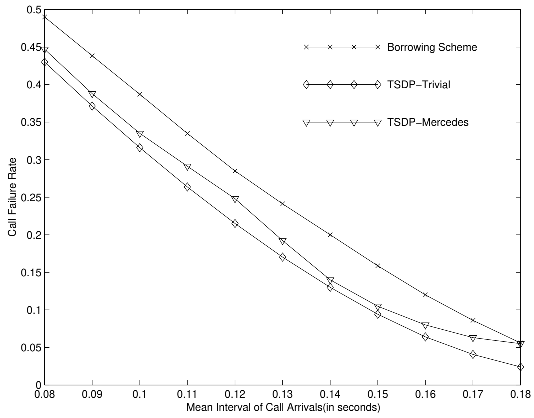
• Distributed Borrowing Scheme — Assume that in most cases it takes only one search to acquire a channel; this assumption is not unrealistic because the cell in search for a channel attempts to collect the most-recent channel usage information before launching a search. The message complexity and acquisition delay for acquiring a primary channel are both zero. The message complexity for acquiring and releasing a secondary channel is $3d^2 + 3d^2 + 3 + 3 + 3$ or $6d^2 + 9$, and the channel acquisition delay is $4T$. If the percentage of secondary channels in the acquired channels is p_s, then the average message complexity and channel acquisition delay are $p_s(6d^2 + 9)$ and $4p_sT$, respectively.

• TSDP with Trivial Information Structure — Assume that on average it takes $n + 1$ searches to successfully acquire a channel, where $n \geq 0$. The message complexity for each channel acquisition and release is $(n + 1) * (3d^2 + 3d^2)$ or $6(n + 1)d^2$, and the acquisition delay is about $2(n + 1)T$.

• TSDP with Mercedes-logo Information Structure — Again, assume that on average it takes $n + 1$ searches to successfully acquire a channel. Then the message complexity for each channel acquisition and release is $(n + 1) * (3d + 3d) + 3d$ or $(6n + 9)d$ and the acquisition delay is $2(n + 1)T$.

In the above analysis, the value of n is most likely to be between 0 and 1. This is because after an unsuccessful search, a cell would have collected the most-recent channel usage information that a second search would succeed with a high probability.

From the above analysis, the "TSDP with trivial-information-structure" scheme requires the highest



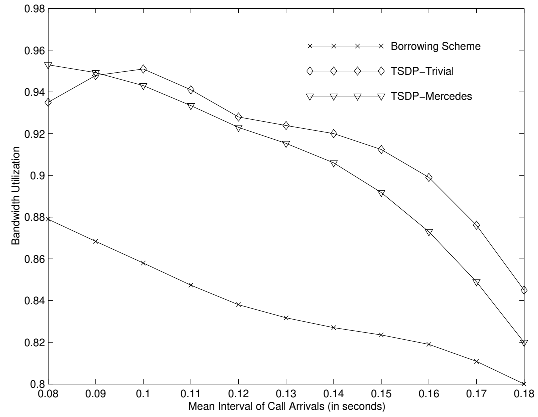


Figure 7: Left: Call Failure Rate. Right: Channel Utilization.

message complexity, while the borrowing scheme has a lower message complexity if most acquired channels are primary channels. However, both are in the order of magnitude of d^2. In comparison, the message complexity of the "TSDP with Mercedes-logo" scheme seems lowest since it is one order of magnitude smaller. This tends to be true if a larger reuse pattern (thus larger d and more cells in a cell's IN) is employed. However, given a smaller reuse pattern such as we used in this simulation, the message complexity of the borrowing scheme could be the lowest.

The wireless network we simulated is composed of 14×14 cells placed in a hexagonal cellular pattern. Only the data observed from the central 10×10 cells are collected and used in the analysis. The MTs moving out of boundary of the 14×14 cell coverage area are assumed terminated. Since the observation focus area is 2 cells away from the boundary, we assume negligible boundary effects on acquired data. There are a total of $N = 392$ channels used as transmission channels in the system. The arrival of new calls is a Poisson process. The lifetime (length) of a new call is decided from an exponential distribution with an average of 180 seconds; its initial service cell is randomly chosen. The stay-time (or sojourn time) of the call in a cell is exponentially distributed with an average of 60 seconds. For a cell c with six neighboring cells n_1, n_2, ..., n_6, the handoff probabilities to them are represented as p_{c,n_1}, p_{c,n_2}, ..., p_{c,n_6}, where $0 \leq p_{c,i} \leq 1$, $i = n_1, n_2, ..., n_6$ and $\sum_{i=1}^{6} p_{c,n_i} = 1$. To simulate different mobility patterns, p_{c,n_1}, p_{c,n_2}, ..., p_{c,n_6} may be set to different values. When a call incurs a handoff, the handoff-to cell is chosen randomly according to the handoff probabilities. An alternative model is to have each individual call associated with a set of probabilities p_{c,n_1}, p_{c,n_2}, ..., p_{c,n_6} which together describe the movement pattern of that mobile user. For instance, if p_{c,n_1} is set to 1 and all other five probabilities set to 0, this would model the case of straight line movement.

In order to simulate a realistic cellular network, we assume non-uniformly distributed traffic in our model. This is implemented by randomly designating variable number of cells as hot cells (or sink cells), which attract both new calls and handoff calls with a randomly generated degree (the probability of a MT being absorbed to a hot cell). These hot cells keep the hot status for an exponentially-distributed period of time and then return to the normal status. Whether there are any hot cells or not is also modeled as an on-off process, with each on/off period exponentially distributed.

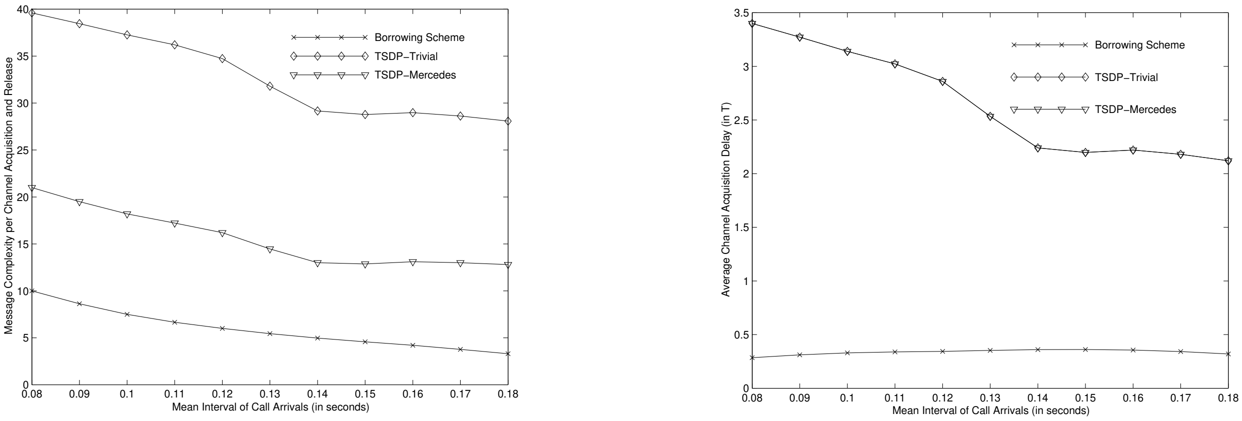


Figure 8: Left: Message Complexity. Right: Channel Acquisition Delay

Figures 7 and 8 compare the performances of the proposed scheme, TSDP-Mercedes, with the other two schemes, TSDP-trivial and Borrowing. The TSDP-trivial scheme has the lowest call failure rate and the highest bandwidth utilization. However, it requires the heaviest overhead, namely the highest message complexity. Although we did not consider the presence of congested communication links between cells in the simulations, the high message complexity could severely invalidate any of the advantages enjoyed by TSDP-trivial, especially when the call arrival rate is high. The channel borrowing scheme, is shown to have the least overhead; however, it has the highest call failure rate and the lowest bandwidth utilization.

The proposed scheme (TSDP-Mercedes) has a performance in-between. It has a low call failure rate and high bandwidth utilization close to that of TSDP-trivial. It also has a relatively low message complexity. Although the channel acquisition delay is higher due to possible failed searches, it is not a concern for new arrived calls. We will show later that by using hybrid search, the channel acquisition delay for handoff calls can be kept low. Both TSDP-Mercedes and TSDP-trivial use the same channel selection strategy, TSDP. But in TSDP-Mercedes, as illustrated in Figure 6, a cell does not have the channel usage information of all cells in its OIN. This simulation result shows that this is not critical for channel selection. Overall, the proposed scheme is more efficient than the other two given that it provides high bandwidth efficiency with low overhead.

Longer acquisition delay is not a big concern for new calls, which are more tolerable to connection delays. However, it is critical to handoff calls. As we proposed a solution to use hybrid search for handoff calls, we simulated hybrid search for handoff calls with two channels being searched for each hybrid search. Figure 9 shows the percentage rate of failed hybrid searches. Compared with the failure rate of sequential searches, the rate is significantly reduced. By adjusting the number of channels to search in hybrid search, the failure rate of hybrid search is expected to be reduced further. As the failure rate of hybrid search is lowered, the average channel acquisition delay is reduced to a bit longer than $2T$, which should be considered tolerable for handoff calls.

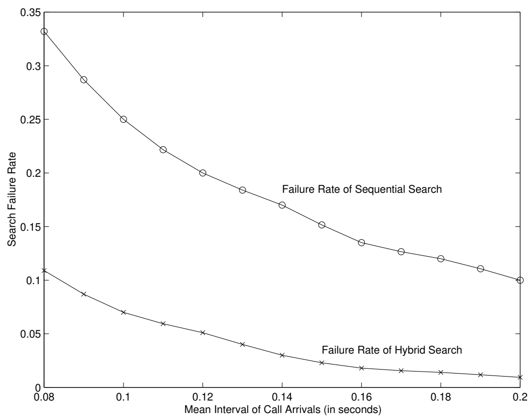


Figure 9: Hybrid Search Failure Rate

9 Conclusion

Distributed dynamic channel allocation, and distributed relaxed mutual exclusion in general, is a difficult problem. We showed that one aspect of the problem is a generalization of the traditional mutual exclusion problem, and decomposed the problem into several subproblems. We developed a general algorithm that guarantees relaxed mutual exclusion for a single channel, proved a necessary and sufficient condition for the information structure, and addressed the issues that arise in distributed channel allocation, including deadlock resolution, dealing with multiple channels, design of efficient information structure, and channel selection strategy. Based on these results, we proposed an example distributed channel allocation scheme. We then compared our scheme with other schemes by simulation study, and showed that it improved the overall performance significantly. We hope that the results reported here will provide a foundation for researchers interested in distributed channel allocation and other relaxed mutual exclusion problems.

We conclude with a pointer to future work. Throughout this paper, our focus has been on channel allocation. Another problem that is important in cellular networks is that of *locating* a mobile phone. In other words, when a call comes in to a particular telephone number, the system as a whole has the responsibility of finding the *current* location of the corresponding phone and forwarding the call to the corresponding base station (which can then forward the call to the mobile phone). This problem too presents challenges somewhat similar to those presented by distributed dynamic channel allocation. In particular, individual sites will not, in general, have up-to-date information about the set of mobile phones currently located in the cells serviced by the various base stations. One approach that has been suggested [21] for dealing with this problem is the 'quorum-based' approach; the idea is that *several* sites will have information about the location of any given mobile phone, so by depending on information from a 'majority' (quorum) of sites, we can reliably locate the phone even if information from some of the sites conflict with each other. Of course, as the mobile phones move above, information about their locations maintained in the various sites would have to be updated in a reasonably timely manner, else even with majority consensus, we will not be able to accurately locate the phones. In any case, it would be interesting to investigate the possibility of combining this type of information with information about channel usage in various cells that we currently maintain.

Further, as messages are exchanged between the cells requesting channels, granting channels, or rejecting requests, we could also piggyback information about the changing locations of the mobile phones that the particular site is aware of, thereby reducing the combined cost of addressing the two important problems of identifying the current locations of mobile phones and of allocating channels to meet current traffic demand.

Acknowledgment

The authors would like to thank the anonymous referees for careful reading of the original manuscript and for providing us detailed comments and suggestions for improvements.

References

[1] A. Baiocchi, F. D. Priscoli, F. Grilli and F. Sestini, "The Geometric Dynamic Channel Allocation as a Practical Strategy in Mobile Networks with Bursty User Mobility", IEEE Trans. Vehicular Tech., pages 14-23, Feb, 1995

[2] G. Cao and M. Singhal, "Distributed Fault-tolerant Channel Allocation for Mobile Cellular Networks," In *IEEE INFOCOM'99*, March 1999, pp. 584-591.

[3] O.S.F. Carvalho and G. Roucairol, "On mutual exclusion in computer networks," *CACM*, Feb. 1983, pp. 146-147.

[4] X. Dong, and T. H. Lai, "Distributed Dynamic Carrier Allocation in Mobile Cellular Networks: Search vs. Update," In *Proceedings of the 17th International Conference on Distributed Computer Systems*, May 1997, pp. 108-115.

[5] X. Dong, and T. H. Lai, "An Efficient Priority-based Dynamic Channel Allocation for Mobile Cellular Networks," In *IEEE INFOCOM'97*.

[6] Y. Furuya and Y. Akaiwa, "A distributed channel allocation scheme for mobile communication systems," *IEICE Trans*. Vol. 74, pp. 1531–37, 1991.

[7] David J. Goodman, *Wireless Personal Communications Systems*, Addison-Wesley, 1997.

[8] C.L. I and P.H. Chao, "Local Packing – Distributed Dynamic Channel Allocation at Cellular Base Station," IEEE Globecom, pp. 293–301, 1993.

[9] C.L. I and P.H. Chao, "Distributed Dynamic Channel Allocation Algorithms with Adjacent Channel Constraints," *PIMRC*, Vol. B2.3, pp. 169–175, 1994.

[10] A. Kahol, S. Khurana, S. K. Gupta, and P. K. Srimani, "Adaptive Distributed Dynamic Channel Allocation for Wireless Networks," In *Proceedings of the 1998 ICCP WORKSHOPS*, August 1998, pp.124-133.

[11] I. Katzela and M. Naghshineh, "Channel Assignment Schemes for Cellular Mobile Telecommunication Systems: A Comprehensive Survey," *IEEE Personal Communications*, June 1996, VOL. 3 No. 3, pp. 10-31.

[12] L. Lamport, "Time, clocks and the ordering of events in a distributed system," *Communications of the ACM*,21(7):558-565, November 1978.

[13] W. C. Y. Lee, *Mobile Cellular Telecommunications: Analog and Digital Systems*, McGraw-Hill, 1995.

[14] N. Lynch, *Distributed Algorithms*, Morgan Kaufmann, 1996.

[15] V. H. MacDonald, "The Cellular Concept," *The Bell System Technical Journal*, Vol. 58, No. 1, January 1979, pp. 15-41.

[16] L. Narayanan and S.M. Shende, "Static Frequency Assignment in Cellular Networks," *Algorithmica*, pages 396-409, 2001.

[17] M. Maekawa, "A $\sqrt{N}$ algorithm for mutual exclusion in decentralized systems," ACM Transactions on Computer Systems Volume 3 , Issue 2 (1985) pp. 145-159

[18] C. Mohan and A. Silberschatz, "Distributed control – Is it always desirable?" *Symposium on Reliability in Distributed Software and Database Systems*, July, 1981.

[19] S. Nishio, K.F. Li, E.G. Manning, "A resilient mutual exclusion algorithm for computer network," *IEEE Trans. on Parallel and Distributed Systems*, July 1990.

[20] R. Prakash, N. Shivaratri, and M. Singhal, "Distributed Dynamic Channel Allocation for Mobile Computing," In *Proceedings of the 14th ACM Symposium on Principles of Distributed Computing*, 1995, pp. 47-56.

[21] R. Prakash and M. Singhal, "Dynamic Hashing + Quorum = Efficient Location Management for Mobile Computing Systems," *Proc. of ACM Symposium on Principles of Distributed Computing (PODC),* 1997, pp. 291–291.

[22] S. Ramanatham, "A unified framework and algorithm for channel assignment in wireless networks," *Wireless Networks*, pp. 81-94, 1999.

[23] G. Ricart and A.K. Agrawala, "An optimal algorithm for mutual exclusion in computer networks," *CACM*, January 1981, 9–17.

[24] B. A. Sanders, "The Information Structure of Distributed Mutual Exclusion Algorithms," In *ACM Transactions on Computer Systems*, Vol. 5, No. 3, August 1987, pp. 284-299.

[25] A. Sen, T. Roxborough, B.P. Sinha, "On an optimal algorithm for channel assignment in cellular networks," *IEEE International Conference on Communications*, vol.2, pp. 1147–1151, 1999.

[26] M. Singhal, "A Taxonomy of Distributed Mutual Exclusion," *Journal of Parallel and Distributed Computing*, 1993, pp. 94-101.

[27] C.W. Sung and W.S. Wong, "Sequential packing algorithm for channel assignment under cochannel and adjacent-channel interference constraint," *IEEE Trans. on Vehicular Technology*, pp. 676–686, Aug. 1997.

[28] I. Suzuki and K. Kasami, "A distributed mutual exclusion algorithm," ACM Trans. on Computer Systems, Nov. 1985.