A Quorum-based Distributed Channel Allocation Algorithm for Mobile Computing Systems

Chakarat Skawratananond and Vijay K. Garg

TR-PDS-1997-010 **July 1997**

Parallel & Distributed Systems group
Department of Electrical & Computer Engineering
University of Texas at Austin
Austin, Texas 78712

A Quorum-based Distributed Channel Allocation Algorithm for Mobile Computing Systems

Chakarat Skawratananond and Vijay K. Garg*
Parallel and Distributed Systems Laboratory
http://maple.ece.utexas.edu
Electrical and Computer Engineering Department
The University of Texas at Austin,
Austin, Texas 78712
{chakarat,vijay}@pine.ece.utexas.edu

Abstract

Since radio spectrum is a scarce resource, efficient allocation of frequency channels is critical for the performance of mobile computing systems. The *update approach* is a way to allocate radio channels among cells in distributed fashion. In update-based algorithms, each cell maintains its local knowledge about channels available for its use by exchanging messages among cells in its interference neighborhood. An advantage of the update approach is its low channel acquisition delay, which is defined as the time between the send event of channel requests and the moment that a channel is successfully acquired. However, the existing update algorithms suffer from high message complexity or high storage overhead. In this paper, we present a distributed update-based algorithm that imposes lower message complexity, while requiring smaller storage overhead than existing algorithms.

Keywords

Distributed channel allocation, mobile computing, distributed network algorithm.

1 Introduction

The radio spectrum is a scarce resource in mobile communication systems. An efficient reuse of the radio spectrum allocated to the system is required as the population of mobile users continue to grow at the tremendous rate. Each mobile host can establish communication with other entities of the network only through its local base station via a wireless channel. Therefore, availability of radio channels plays an important role in achieving good system performance. A major prohibiting factor in radio spectrum reuse is co-channel interference.

*supported in part by the NSF Grants ECS-9414780, CCR-9520540, a TRW faculty assistantship award, a General Motors Fellowship, and an IBM grant.

The radio spectrum is divided into a number of frequency units. If *channels* are those units, then each channel is used to support only one call in a cell at a time. If *carriers* are used as a unit, then the entire bandwidth of a carrier assigned to a cell is time-shared by all the calls initiated within the cell. We choose to use the term channel throughout this paper.

A channel can be reused in a spatially disjoint cell if *reuse constraints* are satisfied. Reuse constraints are conditions used to determine when a channel can be reused by other cells without causing an interference called *co-channel interference*. The reuse constraint we consider is the distance between cells, D_{min}.

To avoid co-channel interference each cell must [3]

1. compute the set of available channels,

2. select one channel from the set of available channels, and

3. acquire the selected channel.

The procedure that performs task (1) and (3) is referred to as the *channel acquisition algorithm*, and the one that performs task (2) referred to as the *channel selection strategy*. Channel selection strategies have been extensively studied in the context of cellular telephone systems. A comprehensive survey on channel selection strategies can be found in [7].

Generally, channel selection strategies can be divided into three different categories: fixed (FS), dynamic (DS), and hybrid (HS). In FS, channels are assigned to each cell according to some fixed reuse pattern so that the co-channel interference is avoided. This strategy is simple, but it does not adapt to the changing traffic conditions and user distribution. To overcome these disadvantages, DS strategies are introduced. Channels in DS can be assigned to any calls as long as there is no co-channel interference. DS provides traffic adaptability at the cost of higher complexity. However, DS is less efficient than FS under high load conditions [7]. HS strategy, a mixture of FS and DS, is introduced to overcome this drawback. Channels in HS are divided into *fixed* and *dynamic* sets. Each cell is assigned with a fixed set as in FCA, and these channels are preferred for use in their respective cells. The dynamic set is shared by all users in the system to increase flexibility.

There are many different ways to perform task (1) and (3). One simple approach is to have each cell requests a channel from a central controller who will ensure that the co-channel interference does not occur. This centralized approach does not scale well, and it has a single point of failure. Another approach is to distribute tasks and responsibilities to each MSS in the system. In this distributed approach, each MSS exchanges information with its neighbors within the co-channel inteference range so that it can make a decision about channel allocation in its cell based on its local knowledge. Therefore, distributed algorithm is more robust and scalable.

A distributed channel allocation algorithm should aim at minimizing the channel acquisition delay.[1] It must also aim at minimizing the amount of information (size and number of messages) that needs to be exchanged per request. Also, the algorithm must be able to cope with failures of the components of the network. One criterion to measure this property is the *failure number* which

[1]The elapsed time between a cell's sending a request message and its acquisition of a channel.

measure the number of cells affected by a faulty support station. Clearly, the goal is to minimize the failure number of the algorithm.

We present a distributed algorithm in which each MSS bears an equal amount of responsibility for channel allocation control. Each cell is required to communicate only with a small subset of its neighbors within the co-channel inteference range in order to acquire a channel. No responses are ever deferred, therefore, the communication set up time is relatively small. The failure number of the algothm is kept at minimum. Moreover, the algorithm is simple to implement, and requires less storage and message overhead than existing algorithms.

The problem of distributed channel allocation can be viewed as a variant of distributed mutual exclusion problem, where each channel corresponds to a critical section. In distributed channel allocation, however, multiple cells can simultaneously use the same channel (enter a critical section at the same time) if the reuse constraint is satisfied. Algorithms previously proposed for channel allocation are more or less based on techniques used in mutual exclusion problems. Next, we briefly describe them, and compare them in detail with the algorithm presented here in the following section.

The paper is organized as follows. The previous work is described in Section 2. Section 3 presents definitions and model used throughout the chapter. System with a specific D_{min} is investigated in Section 4. The technique given in Section 4 is generalized for any D_{min} in Section 5. An optimization of the proposed algorithm is given in Section 7. Experimental results are given in Section 8.

2 Previous Work

Prakash et al. [10] proposed an algorithm based on *deferral* technique used in Ricart and Agrawala [11]. In this algorithm, each MSS decides which channel it can use after gathering information from each cell in its interference neighborhood. Each MSS replies with the information containing its channel use if it is not requesting a channel, or if it is requesting with a larger timestamp than the timestamp of the received request. Otherwise, the response is deferred. With all the received information, the MSS may have to borrow a non-busy channel allocated to its neighbors. With the permission from each neighbor that owns this channel, the MSS can transfer the channel and then uses it to support a communication session. Otherwise, the MSS must select a new channel to transfer, and repeat another borrowing procedure. We refer to this algorithm as PK.

Choy and Singh [2] proposed a new borrowing scheme (referred to as CS) that reduces the number of failed transfers. In this scheme, a randomly selected group of channels is exclusively locked by the MSS before the actual transfer occurs. Each MSS obtains a lock on the selected group of channels by executing an existing dining philosophers algorithm [1]. This ensures that no two interfering MSSs lock the same group of channels simultaneously. This algorithm also guarantees that each lock request is eventually granted. The optimum size of the group varies under different load distributions. It was shown by simulations that CS significantly outperforms PK when the traffic load is high and non-uniformly distributed. This is primarily because pessimistic approach acquires a group of channels, and more requests therefore can be served per each acquisition. This saves a lot of communication and processing overhead

Both PK and CS fall into the *search* category according to [3] since each MSS does not maintain the information about the channel being used by its neighbors. When a channel is needed, the MSS searches all neighboring cells to compute the set of currently available channels. Due to deferring mechanism used to resolve conflicts, acquisition delay in search algorithms will significantly increase as traffic in the network grows. In particular, the response of the request message received by each cell will be deferred even when the cell is requesting a different channel.

Instead of gathering information each time a channel is needed, each MSS could maintain a set of available[2] channels by informing cells in its interference neighborhood each time it acquires and releases a channel. This is called *update* scheme [3]. A conflict occurs if two cells within each other interference neighborhood concurrently request the same channel. Therefore, to request a channel, each cell must send a request message to each cell in its interference neighborhood. The request is successful if each response is a *grant* message. This is known as the *basic update* scheme. This scheme reduces acquisition delay at the expense of higher message complexity. Algorithms presented in [5, 6] take this approach.

Dong and Lai [3] proposed an update algorithm (referred to as DL). They reduced message complexity by using an *inquire-some mutual exclusion model.* This algorithm is designed to work with a class of HCA strategies such that carriers[3] assigned to each cell with different priorities. In particular, each cell is initially given a set of preferred carriers so that they are always selected whenever available before other carriers. This priori information about carrier status is referred to as *channel reuse planning.* To request a carrier, each cell sends request messages only to a small subset of its interference neighborhood, depending on the carrier being requested. Thus, each cell has to maintain a list of cells for each carrier to which request messages must be sent.

Garg et al. [4] proposed two update algorithms. In this paper, the problem of channel allocation is modelled as a generalization of the list coloring problem. Each cell i is initially associated with a group of channels (*list*) from which the channels to support communication sessions in cell i must be chosen. Each support station is also assigned with a priority number (color) such that no two support stations within each other interference neighborhood have the same color. Between neighboring support stations that request channels at the same time, the one with the smaller color chooses first. To avoid starvation, the synchronization mechanism of Choy and Singh [1] is employed as in CS. Again, the response time of this algorithm is high due to the fact that a cell can be blocked by its neighbors if they are requesting simultaneously even though the requested channels are different. This algorithm is referred to as G-Det. The second algorithm is for the system where the size of the *list* is much larger than the total numbers of channels needed by the cell. In this algorithm (referred to as G-Ran), a group of channels are randomly picked from its list, and each support station must ensure that the selected frequencies are not currently used by its neighbors. This goal is accomplished using message passing among support stations.

Our algorithm (referred to as QB) uses Maekawa's technique [9] to reduce message complexity in basic update algorithm. Similar to DL, request messages are sent to only a small subset of cells

[2]Channels that cells can use without co-channel interference

[3]*Carrier* is the basic unit of resource used in their algorithm.

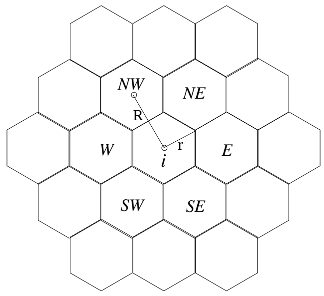


Figure 1: Hexagonal Cellular Model

in the interference neighborhood. However, this subset does not depend on the requested channels. This algorithm therefore requires less memory space, and is simpler to implement. Unlike DL, this algorithm can be used with any channel reuse plannings. Moreover, the generalization of the update algorithm to the system where the interfering cells is more than one hop away is also presented.

3 Model

It is assumed that the allocated radio spectrum is divided into a number of channels. A unique ID is assigned to each channel. Each cell also has a unique ID represented by a pair (x, y), the position of cell on a 2-dimensional plane. Each cell i is logically modelled as a hexagon of radius r; therefore, it has six neighbors, denoted by $E_i, W_i, NE_i, NW_i, SE_i, SW_i$, according to its direction from i. Figure 1 displays a cellular network model.

The distance between cells is defined as the distance between the two centers. If R is the distance between two neighboring cells, then $R = \sqrt{3}r$. A channel can be reused in any two cells if the distance between them is at least D_{min}. We say that two cells are in each other *interference neighborhood* if the distance between them is less than D_{min}. Note that D_{min} must be greater than R.

We also assume that all MSSs are connected and every message sent between them is eventually received in FIFO order. Each MSS maintains Lamport's logical clock [8] so that events at different MSSs are totally ordered.

Given a cell c, $IN(c)$ denote the set of all the cells whose distance to c is less than D_{min}.

Definition 3.1

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

where $dist(c, c')$ denotes the distance between c and c'. We say that two cells, i, j, interfere with each other if the distance between them is less than D_{min}. We also say that i is in the interference neighborhood of j if $i \in IN(j)$.

The goal of any channel allocation algorithm is to ensure that no two interfering cells simultaneously use the same channel. To achieve this goal, any two requests from two interfering cells must be known to at least one of the arbitrators. If we assume that cell i obtains a permission from each member of a set S_i (*request set*) such that $S_i \subset IN(i)$, there must be at least one common cell between a pair of S_i and S_j for any two interfering cells i, j. We refer to this property as the pairwise nonnull intersection property ($\mathcal{PN}$).

We next present a scheme to construct the request set such that $\mathcal{PN}$ is satisfied for a given D_{min}.

4 3-Cell Cluster System

In this system, a channel cannot be reused within the same cell or the neighboring cells. This is because $R < D_{min} \leq \sqrt{3}R$. Consequently, there are exactly 6 members in each cell's interference neighborhood. To satisfy $\mathcal{PN}$, we assign to each cell i the following request set: $S_i = \{i, NW_i, SW_i, E_i\}$. We call this 1-*hop request set.* It is easy to see that the following property can be derived from 1-hop request set. Let S_{ij} denote $S_i \cap S_j$.

Property 4.1 *For any two interfering cells i, j*

- $|S_{ij}| = 1$, *and*
- $j \in S_{ij}$ *or* $i \in S_{ij}$

Next we present a distributed channel allocation algorithm for 3-cell cluster system using 1-hop request set.

4.1 The Algorithm

We now provide an informal description of the 3-cell cluster algorithm shown in Figure 2. Let $\mathcal{C}$ denote the set of channels shared by each cell in the system. Each cell c maintains two sets, U_c and I_c. U_c is the set of channels currently used by c. I_c is the set of interfered channels at c. Initially, every channel $k \in \mathcal{C}$ is available for use by c. A channel becomes an *interfered* channel for c if it is acquired by a cell in $IN(c)$.

When a cell needs a channel to support a communication session, it selects a channel k from $(\mathcal{C} - (I_c \cup U_c))$ based on the underlying channel allocation strategy. If the underlying strategy is HCA, then channels in the fixed set are used whenever it is available. To acquire k, a cell sends a request message ($REQ(k)$) to each cell in its request set except itself. Once c receives a grant ($GRNT(k)$) from every cell in its request set, and if at this moment k does not belong to I_c, then channel k can be used to support a communication session in cell c, and is added into U_c. Next, c sends an acquisition message ($ACQ(k)$) to each cell in its interference neighborhood except those in its request set to inform about its use of channel k. When a cell receives $ACQ(k)$ message, k is added into its set I.

On receiving a request for channel k', c replies with a reject message (REJ) if either c is using this channel ($k' \in U_c$) or c is also requesting k' with a smaller timestamp. Otherwise, c replies with a grant message ($GRNT$), and k' is added into I_c. Note that k' is added into I_c immediately after

```
MSS_i::
Var
      I_i:  the set of interfered channels at i.
      U_i:  the set of used channels at i.
      S_i:  the request set of i.
      C:  the set of channels shared by all cells.
```

(A) <u>To support a communication session</u>
  MSS$_i$ selects a channel $k \in (\mathcal{C} - (I_i \cup U_i))$, according to the
  underlying allocation strategy, and sends a $REQ(k)$ to each cell in S$_i$

(B) <u>Upon receiving a $REQ(k)$ from cell j</u>
  MSS$_i$ replies with a $REJ(k)$ if either $k \in U_i$ or
  MSS$_i$ is also requesting for k with a smaller timestamp;
  otherwise, it replies with a $GRNT(k)$ and add k into I_i

(C) <u>The request for channel k is successful if</u>
  (C1) Every response is a grant, and
  (C2) $k \notin I_i$

(D) <u>Upon the request for channel k is successful</u>
  send $ACQ(k)$ to each cell $c \in IN_i - S_i$
  add k in U_i

(E) <u>Upon receiving a $REJ(k)$ from j or conditions (C1) and (C2) are false</u>
  send a $REL(k)$ to each cell $c \in S_i$ except j, and go back to (A)

(F) <u>Upon a communication session using channel k terminates</u>,
  MSS$_i$ sends a $REL\_TERM(k)$ to each cell in IN_i

(G) <u>Upon receiving a $REL(k)$ or $REL\_TERM(k)$</u>
  remove k from I_i

(H) <u>Upon receiving a $ACQ(k)$</u>
  add k into I_i

Figure 2: The 3-cell Cluster Algorithm

c has granted the request for k'. When the communication session using k is terminated, c sends a REL_TERM message to each cell in I_c.

A cell realizes that its acquisition of channel k is failed when (1) a $REJ(k)$ has been received, or (2) a grant has been received from each cell in S_c but $k \in I_c$. Once the acquisition of channel k is failed, c sends release messages ($REL(k)$) to those cells that have already granted k to c. On receiving $REL(k)$ or $REL\_TERM(k)$, k is removed from I_c.

Once an acquisition is failed, it means that a *conflict* has occurred. A conflict occurs when there are at least two cells in each other's interference neighborhood concurrently request the same channel.

4.2 Correctness Proof

Theorem 4.2 *(Safety) No two cells within each other's interference area concurrently use the same channel.*

Proof: Let there be two cells i, j within each other interference area. We show that i and j cannot simultaneously use the same channel k. Without loss of generality, we assume that the timestamp of i's request is smaller than that of j. Due to Property 4.1, there are only 2 cases to be considered.

- $j \in S_i$: If i is using k, k must be in I_j and then j will not be able to use k unless i releases k. Similarly, if j is using k, then i will receive a REJ message from j. Therefore, i's request will fail. If both i, j are in the process of requesting k, i must receive a grant from j since its timestamp is smaller than j's. Therefore, j's request will fail even though it receives $GRNT(k)$ responses from other cells in S_j.

- $i \in S_j$: If either i or j is using k, then the other cannot use k by the same argument as in the first case. If both i and j are in the process of requesting k, then j will receive a $REJ(k)$ message from i because i's timestamp is smaller than j's.

□

Theorem 4.3 *(Concurrency) The requests of a channel k from any two non-interfering cells will be granted.*

Proof: When a cell c receives a request for channel k, c decides to send $GRNT$ or REJ message by only determining its U_c. Therefore, any two non-interfering cells requesting for the same channel k will successfully acquire k. □

Theorem 4.4 *(Deadlock freedom) The algorithm is deadlock-free.*

Proof: In this algorithm, each cell may either reject or grant a request message, but it never defers its response. Thus, the algorithm is deadlock-free. □

We next prove that *livelock* never occur in the 3-cell cluster algorithm. We call an event livelock if more than one cells within each other's interference neighborhood compete for the same channel, and each one of them fails to acquire the channel.

Theorem 4.5 *(Progress) A conflict on any channel k is always resolved in such a way that there is exactly one successful acquisition.*

Proof: We need to prove that if there are x cells within each other's interference neighborhood requesting for a channel k, then there will be exactly one cell c successfully acquiring k.

Since $R < D_{min} \leq \sqrt{3}R$, it is easy to see that the only possible value of x is either 2 or 3. For $x = 2$, the theorem follows from the safety property given in Theorem 4.2. For $x = 3$, assume that cells c_1, c_2, c_3 are within each other's interference neighborhood and requesting a channel k with timestamps t_1, t_2, t_3, respectively. Without loss of generality, we assume that $t_1 < t_2 < t_3$. Let

$p \rhd q$ be an abbreviation for *"a request from p is received by q"*. It follows from the configuration of 1-hop request set that either (1) $c_1 \rhd c_2$, $c_2 \rhd c_3$, and $c_3 \rhd c_1$, or (2) $c_1 \rhd c_3$, $c_3 \rhd c_2$, and $c_2 \rhd c_1$.

In (1), c_3's request will be rejected by c_1 since $t_3 > t_1$. c_2's request will fail because c_2 receives the request from c_1 which has a lower timestamp. Hence, only c_1's acquisition will be successful.

In (2), c_3's request will be rejected by c_2 since $t_3 > t_2$. c_2's request will be rejected by c_1 since $t_2 > t_1$. Again, only c_1's acquisition will be successful. □

4.3 Performance Analysis

We now discuss message complexity and acquisition delay incurred by the 3-cell cluster algorithm. Let n_s denote the number of request messages sent per acquisition, and n_{if} denote the size of $IN(c)$ for each cell c. The number of messages required for a successful acquisition (excluding ACQ and REL_TERM) is $2 * n_s = 6$. For a failed acquisition, the number of messages required is at most $3 * n_s$. Therefore, if a cell has to make m attempts before it finally acquires a channel, the number of messages required per a channel acquisition is at most $2 * n_s + 3 * n_s * (m-1) + (n_{if} - n_s) + n_{if}$. The summation of $(n_{if} - n_s)$ and n_s is the total number of ACQ and REL_TERM messages. After substituting n_s and n_{if} with 3 and 6, respectively, the number of messages required is at most $9(m-1) + 15$. In the basic update scheme [5, 6], the number of messages required per a channel acquisition is exactly $12(m-1) + 18$.

Let D denote the average communication delay between MSSs. The average acquisition delay for the algorithm is $2 * m * D$. The failure number of the 3-cell cluster algorithm is 3 since each support station receives request messages from exactly three neighboring cells.

Message complexity and acquisition delay of the 3-cell cluster algorithm are the same as DL. In contrast, progress is not always guaranteed in DL.

5 Generalization

In this section, we present a general scheme to construct a request set for a given D_{min}. We also propose a distributed channel allocation algorithm to use in association with those request sets.

Let r_i^j denote the set of cells that are j hops away from i in the east, northwest, and southwest directions relative to i. The *n-hop* request set of cell i (denoted by S_i^n) is defined as follows:

$$S_i^n = i \cup r_i^1 \cup \ldots \cup r_i^n$$

Figure 3 shows S_i and examples of S_{ij} for 2-hop request set. In the previous section, we have shown that 1-hop request set satisfies $\mathcal{PN}$ when $R < D_{min} \leq \sqrt{3}R$. The following lemma shows that 1-hop request set also satisfies $\mathcal{PN}$ even when $R < D_{min} \leq 2R$.

Lemma 5.1 *The* 1*-hop request set satisfies* $\mathcal{PN}$ *if* $R < D_{min} \leq 2R$

Proof: Since $R < D_{min} \leq 2R$, $R \leq dist(c', c) < 2R$ for any $c \in IN(c')$. From the configuration of 1-hop request set, it is easy to see that $S_c \cap S_{c'} \neq \emptyset$ □

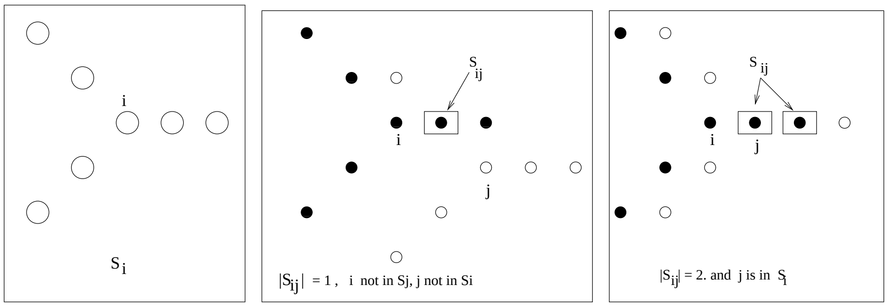


Figure 3: S_i and S_{ij} in 2-hop request set

Theorem 5.2 *For any integer $n \geq 1$, the n-hop request set satisfies $\mathcal{PN}$ if $nR < D_{min} \leq (n+1)R$*

Proof: Please see Appendix. □

The following property can be derived from the n-hop request set's configuration:

Property 5.3 *For any two interfering cells i, j,*

- *If $|S_{ij}| \geq 2$, then either $i \in S_j$ or $j \in S_i$.*

5.1 The Generalized Algorithm

We now provide an informal description of the generalized algorithm shown in Figure 4. This algorithm is intended for the system with $D_{min} > \sqrt{3}R$. In Figure 4, we use $*$ to denote *don't care*.

The definition of U_i and $\mathcal{C}$ here are the same as in Section 4. The request set S_i is determined by D_{min}. The set I_i is a set of *tickets*. A ticket is a *struct* containing four fields: $Status, Cell\_id, Ch\_id, TS$. Whenever cell i grants a request of channel k with timestamp ts from cell j, a *ticket* (UC, j, k, ts) must be added into I_i (UC for *unconfirmed*). $Status$ is a boolean variable, and becomes CF (*confirmed*) when i receives an $ACQ(k, ts)$ from j. An $ACQ(k, ts)$ is sent to each cell in the interference neighbood if the acquisition of channel k is successful.

Unlike the previous algorithm, the response of each request can be either $GRNT$, C_GRNT, and REJ (Step `B`). A cell replies with C_GRNT to the request from cell j for a channel k if it has given at least one grant to another request from the cell within the interference neighborhood of j with a bigger timestamp. The ID's of those cells are also sent along with C_GRNT. A cell replies a channel k's request from cell j with $GRNT$ if either it is not using k, or it is not requesting k with a smaller timestamp, or it never grants the request of k to any cells that interfere with j before. Otherwise, the cell replies with REJ.

A cell determines if its request is successful by checking all three conditions given in Step `C`. If i receives $C\_GRNT(W, k)$ from p, it means that the request from each cell $w \in W$ have been granted by p and the request timestamp of w is greater than that of i. If $w \in S_i$ or $i \in S_w$, then w will receive i's request or i will receive w's request, respectively. In either case, w's request will be failed given that i's request is not rejected by w. Checking if $i \in S_w$ can be done easily by calculating the distance and direction between i and w.

Once a cell realizes that its request has been rejected, it sends REL message to each cell in its request set except the cell(s) from which REJ message has been received. After a call has been terminated in cell i, REL_TERM is sent to each cell in IN_i. On receiving ACQ, REL or REL_TERM, set I_i is updated accordingly. Note that the algorithm presented in Section 4 is a special case of the one presented here.

5.2 Correctness Proof

Theorem 5.4 *(Safety) No two cells within each other's interference neighborhood simultaneously use the same channel.*

Proof: Please see Appendix. □

Theorem 5.5 *(Concurrency) Two requests from two non-interfering cells for the same channel will be granted.*

Proof: Consider any two cells x, y such that $x \notin IN(y)$, and cell z such that z is in the interference neighborhood of both x and y. Assume that z has granted the request of channel k from x. When cell z receives the request from y, and z is not using and not requesting a channel k, y's request will be granted (Step (B)) even though $\exists t \in I_z : t.ChID = k$ since $x \notin IN(y)$. □

Theorem 5.6 *(Deadlock freedom) The generalized algorithm is deadlock-free.*

Proof: Each cell may either send REJ or $GRNT$ or C_GRNT, but it never defers its response. Thus, the algorithm is deadlock-free. □

5.3 Performance Analysis

We first describe the performance of the algorithm in term of message complexity and acquisition delay. Then, we give the comparison with the previous works.

Let n_s , n_{if}, and D be defined as in Section 4. If there is no conflict, the number of messages per channel acquisition is $2 * n_s + 2 * n_{if}$ which is the sum of n_s REQ messages, n_s $GRNT$ or C_GRNT messages, n_{if} ACQ messages, and n_{if} REL_TERM messages. The average delay is $2 * D$.

If there are conflicts, and a cell takes m requests before it acquires a channel. The average delay is $2 * D * m$. The number of messages becomes $2 * n_s + 3 * (m - 1) * n_s + 2 * n_{if}$. It is easy to see that the generalized algorithm has n_s failure number.

6 Discussion

In this section, the detailed comparison between the quorum-based algorithms and existing search and update algorithms is given.

Comparison with search algorithms

$\texttt{MSS}_i$::

Var

struct *ticket* {

$Status$: boolean (CF or UC)

$Cell\_id$: cell ID

Ch_id : channel ID

TS : timestamp }

I_i : a set of *ticket*

$U_i, S_i, \mathcal{C}$: as defined in Figure 2.

(A) <u>To support a communication session</u>

(A1) MSS_i selects a channel k from $\mathcal{C}$ such that $k \notin U_i$ and $t.Ch\_id \neq k, \forall t \in I_i$.

(A2) sends a $REQ(k)$ to each cell in S_i

(B) <u>Upon receiving a $REQ(k)$ with timestamp TS from cell j</u>

(B1) MSS_i replies with a $REJ(k)$ if

$k \in U_i$ or

MSS_i is also requesting for k with a smaller timestamp, or

$\exists(UC, j', k, ts) \in I_i : ts < TS$ and $j' \in IN(j)$,or

$\exists(CF, j', k, *) \in I_i$ and $j' \in IN(j)$

(B2) MSS_i replies with a $C\_GRNT(W, k)$

if $W \neq \emptyset$, where $W = \{j'/(UC, j', k, ts) \in I_i \wedge ts > TS \wedge j' \in IN(j)\}$

(B3) otherwise, it replies with a $GRNT(k)$ and add (UC, j, k, TS) into I_i

(C) <u>Cell i's request of channel k is successful if</u>

(C1) Every response is either a $GRNT(k)$ or $C\_GRNT(CID, k)$,and

(C2) For each $C\_GRNT(W, k)$ message that cell i receives,

$\forall w \in W : w \in S_i$ or $i \in S_w$, and

(C3) $\nexists t \in I_i : t.Ch\_id = k$

(D) <u>Upon the request of channel k with timestamp t is successful</u>

(D1) cell i sends an $ACQ(k)$ message to each cell in IN_i, and add k in U_i

(E) <u>Upon conditions in (C) are false or a $REJ(k)$ has been received</u>

(E1) send a $REL(k)$ to each cell in S_i except the one(s) rejecting the request, and go back to (A)

(F) <u>Upon receipt of $ACQ(k)$ from cell j</u>

(F1) if $(UC, j, k, *) \in I_i$ then turn UC to CF, otherwise, add $(CF, j, k, *)$ into I_i

(G) <u>Upon receipt of $REL(k)$ or $REL\_TERM(k)$ from j</u>

(G1) remove the corresponding *ticket* from I_i

(H) <u>Upon the call using k is terminated</u>

(H1) send $REL\_TERM(k)$ to each cell in IN_i

Figure 4: The Generalized Algorithm

As mentioned earlier, search algorithms in general are more likely to suffer from the high acquisition delay during the high traffic load. In PK, the number of messages required per acquisition is smaller than that of update algorithms. The exchange of messages ($O(n_{if})$) is needed only when the currently allocated channels to a cell are not sufficient to support its call requests. Therefore, under light traffic load, message exchange is not necessary. However, during the high traffic load, the size of the response messages is larger than that of update algorithms. This is because response messages must carry the information about channels being used, allocated, or transfered in the neighboring cells as opposed to grant or reject in the update algorithms.

CS also suffers from the high acquisition delay under the high traffic load. When all the allocated channels in a cell are occupied, messages ($O(n_{if})$) are exchanged among the interfering support stations to avoid co-channel interference and starvation.

In the 3-Cell cluster system, the failure numbers of PK and CS is $|IN(c)| = 6$. This is because a request message is sent to each interfering neighbor. The simulation results that compares acquisition delays of the quorum-based algorithm and search algorithms are given in the following section.

Comparison with update algorithms

Here, we give a detailed comparison between the proposed algorithm and Garg et al. In G-Det, a cell has to wait for all its neighbors to select channels in the worst case scenario [4]. G-Det has failure number $|IN(c)|$. The synchronization mechanism used in G-Ran is similar to the one used in QB. Response messages are never deferred in G-Ran. However, the number of request messages used in QB is one-half of that used in G-Ran (in the 3-Cell cluster system). It is less than one-half in the system with arbitrary D_{min}. The failure number of G-Ran is $|IN(c)|$.

Here, the detailed comparison between QB and DL is as follows.

- The request set used in DL depends on the carrier [4] being requested. Therefore, each MSS needs additional memory space to keep a set of cell addresses for each carrier. On the other hand, each cell in our algorithm is required to maintain only one request set. Moreover, each cell in DL needs to know $IN(c)$ for any cell c from which it receives a request message. We only require that each cell can determine if $i \in IN(j)$ and $i \in S_j$ for any cells i, j. Hence, the memory overhead required in their algorithm is higher than that required in ours.
- The number of request messages used in DL is generally smaller than ours. The following table shows the size of the request set used in their algorithm and ours for $D_{min} = \sqrt{3}R, 2R, \sqrt{7}R, 3R$.

Algorithm \ D_{min}	$\sqrt{3}R$	$2R$	$\sqrt{7}R$	$3R$
DL	3	4	3	3 or 4
QB	3	3	6	6

Nevertheless, our request messages in the n-hop request set are only sent to cells at most n hops away in exactly three directions: northwest, southwest, and east. Therefore, a sender could send only three request messages to its neighboring cells in those directions, and let

[4]In DL, a carrier is a basic unit of resource allocation , therefore, their 'carrier' is equivalent to 'channel' in our context

each one of them forward the request message to the next destination. As a result, the routing for our request messages is simple, and the size of message overhead is smaller since only the direction and the number of hops are required.

- The failure number of DL is $|IN(c)|$. This is because each cell c must respond to the request message sent from another cell $j \in IN(c)$ if j is requesting a channel r, the primary channel of c. The failure number of QB is n_s.

- The success conditions[5] in [3] include checking whether some messages have been received. In particular, for each $C\_GRNT(W, k)$ message received by a cell c and for each $w \in W$, c must have already received a $GRNT(k)$ message from some cell p in the intersection of c's request set and w's interference neighborhood. Therefore, it is possible that a channel request can be considered failed only because of the transmission delay caused by the underlying network. On the other hand, the success conditions in our algorithm is static. To determine if a request is successful, each cell only uses its local knowledge about positions of cells within the system (Step `(C2)`). Therefore, the request in our algorithm will never fail because of the delay imposed by the wired network.

- DL is designed to work with a specific channel reuse pattern. On the other hand, our algorithm can be used with any dynamic or hybrid allocation strategies including the one used in their algorithm.

Livelock

In the generalized algorithm, it is possible that $k \geq 3$ cells in each other's interference neighborhood request for the same available channel, but none of them will succeed. We call this *livelock* since each of these k cells need to make another request attempt even though there is a channel that can be used without interference. Livelock in the generalized algorithm can be illustrated by the following scenario. Let there be three cells c_1, c_2, c_3 in the system with $D_{min} = 2R$ request for the same channel k with timestamps $t_1 < t_2 < t_3$, and these three cells are located in such a pattern depicted in Figure 5. If c_1's request arrives at cell c_4 after c_3's request, then c_1 will receive a $C\_GRNT(3, k)$. As a result, c_1's request will be failed because $c_1 \notin S_3$ and $c_3 \notin S_1$. c_2's request also fails since it is rejected by c_1. c_3's request also fails even though it is granted by every member of its request set if the receipt of the last $GRNT$ message occurs between the receipt of REQ and REL messages from c_2.

Note that this livelock scenario will not occur in our 3-cell cluster algorithm as shown in Theorem 4.5. Livelock can also occur in DL even when $R < D_{min} \leq \sqrt{3}R$. Similarly, the algorithm proposed by Prakash et al. also suffer from livelock. An example of livelock scenario occurring in Prakash et al.'s algorithm was given in [2].

It will be seen from the experimental result given in the following section that the possibility for livelock to occur is extremely low. It will be even lower if a conflict reduction scheme[6] is applied.

[5] the conditions each cell uses to determines whether or not a request is successful (Step `(C)` in Figure 4).

[6] A scheme that reduces the chance for cells within each other's interference neighborhood to select the same channel to request

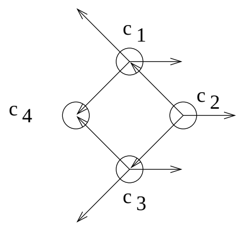


Figure 5: An example of livelock in 1-hop request set

Therefore, livelock will have a very small impact on the performance of the proposed algorithms.

Starvation

In theory, it could occur in every algorithm except the one proposed by Choy and Singh that some cells may experience consecutive request failures or never be able to acquire a channel to support a call (even though the channel being requested is changed every time the cell makes another request attempt). However, as in livelock, the possibility for this to occur will be extremely low.

7 Enhancement

A common weakness of update-based algorithms is high message complexity due to *acquisition* and *release* messages sent to each cell in the interference neighborhood. Given a cell c, the size of $IN(c)$ grows rapidly as D_{min} increases. For instance, if $D_{min} = 3R$, $|IN(c)| = 30$. We here propose a scheme to further reduce the number of messages required in our algorithms at the expense of higher acquisition delay.

We can reduce message complexity in the generalized algorithm by sending ACQ and REL_TERM messages only to S_c rather than $IN(c)$, and in the 3-cell cluster algorithm by omitting ACQ messages and sending REL_TERM messages only to S_c. It is easy to see that the modified algorithm still guarantee the safety property, that is, no two cells within each other's interference neighborhood will concurrently use the same channel. However, the number of conflicts will be higher since the chance that two cells within each other's interference neighborhood request the same channel is higher. With the proposed scheme, only cells in the request set of a cell c and c itself have knowledge of channels currently used by c. Hence, any cell in $IN(c) - S_c$ can request for a channel currently used by c, and later will be rejected by c. However, this will rarely occur if the traffic load is relatively low and/or the allocation strategy reducing the chance of selecting the same channel by interfering cells is applied. We conduct an experiment to investigate the increase of conflicts caused by this scheme, and the results are shown in Section 8.

As a result, the number of messages required per acquisition in the modified 3-cell cluster algorithm becomes $2n_s + 3n_s(m-1) + n_s$. The number of messages required per acquisition in the modified generalized algorithm becomes $2n_s + 3n_s(m-1) + n_s + n_s$.

Since the number of request attempts is likely to be greater than 1 in the modified algorithm under the heavy traffic and equal to 1 in the original algorithm, it is important to calculate the value of m causing the number of messages required in the modified algorithm exceeds the number

of those in the original algorithm with $m = 1$. If we let m^* denote the value of m when the number of messages required per acquistion in the modified algorithm is equal to the number of those in the original algorithm with $m = 1$.
For both 3-cell cluster and generalized algorithms,

$$m^* = \frac{2}{3}\left(\frac{n_{if}}{n_s} - 1\right) + 1$$

Once the number of request attempts in the modified algorithm is greater than m^*, the modified algorithm is outperformed by the original algorithm in both message complexity and acquisition delay. For example, $m^* = 3$ when $D_{min} = 2R$, and $m^* = \frac{5}{3}$ when $D_{min} = \sqrt{3}R$.

8 Experimentation

8.1 Simulation Environment

The system under simulation is a wrapped-around layout with 7×7 cells. There are totally 100 channels shared by all 49 cells. Each cell randomly selects a channel from its available set with the same priority. No special channel selection scheme is performed to prevent cells in each other's interference neighborhood request for the same channel simultaneously. If a request attempt is failed, the cell randomly selects another channel.

We assume that the average one-way communication delay between any two MSSs is 2 milliseconds. The duration of a communication session during which a channel is in use is exponentially distributed with mean 120 seconds. Under uniform traffic distribution, the arrival rate of channel request in each cell has a Poisson distribution with mean λ. Under non-uniform traffic distribution, the arrival process in each cell can be in either *normal* or *hot* state. This model of non-uniform traffic distribution was also used in [3]. The arrival rate in hot state is 3 times higher than normal state. The period of being in hot and normal state are exponentially distributed with mean 180 and 1800 sec., respectively. Handoff calls[7] and new calls are processed with the same priority.

Each run processes 98,000 calls, but data was collected after 49,000 calls had been processed in order to eliminate the impact of startup transients. The number of request attempts (m) for each call is collected, and the average value of m is calculated for each run. The mean value of five such runs, each with a different random number seed, corresponds to a single data in the figures.

We run our simulation on two systems: $D_{min} = \sqrt{3}R$ with the 3-cell cluster algorithm and $D_{min} = 2R$ with the generalized algorithm. Each system is also run with the corresponding modified version for comparison with its original counterpart.

8.2 Experimental Results

Comparison with update algorithms:
Figure 6 and 7 respectively illustrate the average number of request attempts per call under uniform and non-uniform traffic distributions.

[7]the ongoing calls which are transferred from one cell to another due to the mobility of MHs

(a) (b)

Figure 6: The result for uniform traffic distribution. (a) $D_{min} = \sqrt{3}R$, (b) $D_{min} = 2R$

(a) (b)

Figure 7: The result for non-uniform traffic distribution. (a) $D_{min} = \sqrt{3}R$, (b) $D_{min} = 2R$

The results demonstrate that the average number of request attempts (m) in the system using the original algorithm under both uniform and non-uniform traffic distributions is very close to 1 (even though it appears in the graphs that it is always 1). For comparison purposes, we could assume that the result from DL would be similar to the result from our original algorithm. This is because every cell in their algorithm also informs each cell in its interference neighborhood each time it acquires a channel.

Under the modified algorithm, m increases as the traffic load grows; the rate at which m increases in $D_{min} = \sqrt{3}R$ is less than that of $D_{min} = 2R$. This is because the size of the interference set , $IN(c)$, for each cell c in $D_{min} = \sqrt{3}R$ is smaller than $D_{min} = 2R$. The number of request attempts in the uniform traffic distribution increases at the lower rate than that of non-uniform distribution. However, this higher acquisition delay can be reduced by allocating to each cell a set of preferred channels as in the hybrid channel allocation strategy. In addition to preferred channels, we can further reduce conflicts by applying the allocation strategy that prevents interfering cells to select the same channel.

In Figure 8 and 9, we compare the number of messages used per acquisition in our algorithms with the algorithm proposed by Dong and Lai. Each point in the graphs of our algorithms is calculated from the corresponding data in Figure 6 and 7. For DL, each point is calculated from the formula presented in [3] given that each cell has no preferred channels and the average number of request attempts per call is 1. The results demonstrate that the modified algorithm requires less messages than both the original algorithm and DL. It remains lower until the channel demand reachs the point where the number of request attempts per call in the modified algorithm is equal

(a) (b)

Figure 8: The number of messages required per acquisition for $D_{min} = \sqrt{3}R$. (a) uniform distribution, (b) non-uniform distribution

(a) (b)

Figure 9: The number of messages required per acquisition for $D_{min} = 2R$. (a) uniform distribution, (b) non-uniform distribution

to m^*.

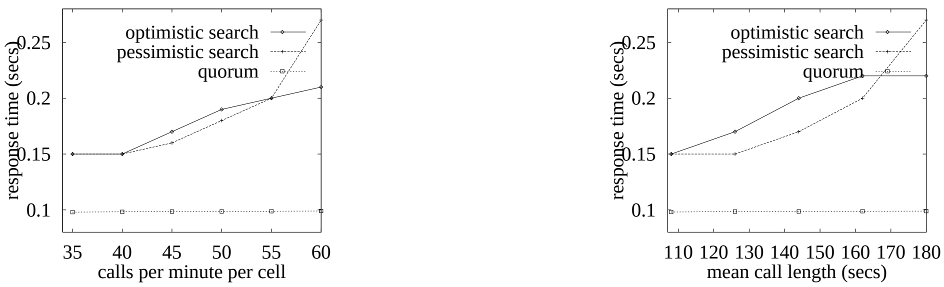
Recall that the average acquisition delay is $2 * m * D$, where D is the average communication delay between support stations. Therefore, if D is small (which is true in the existing underlying networks), the impact of the slight increase of m on the acquisition delay should be negligible. As a result, message complexity in the update-based algorithms can be reduced at a slight increase in acquisition delay by using our modified algorithm.

Network controllers can employ our modified algorithm to significantly reduce traffic load in the underlying wired network at a slight cost of acquisition delay. For example, each cell in the system can initially be set to use the modified algorithm. The heavily loaded cells will be controlled to switch to the original algorithm when the average number of request attempts m in the cell exceeds a predetermined threshold. However, this threshold must be less than m^* given in Section 7. The controller can also switch any cell back to its initial mode whenever its number of request attempts falls below the threshold again.

Comparison with search algorithms:

Here, we show the results from the simulation comparing the quorum-based algorithm with search algorithms: PK and CS.

The experimental setup used here is similar to the above except that there are 400 channels in the system: 302 channels are shared by all cells, and 2 channels are exclusively allocated to each cell. One-way communication delay is 100 ms. Only the 3-Cell cluster system is considered. For the



(a) (b)

Figure 10: The acquisition delay under uniform load distribution. (a) effect of increasing mean arrival rate, (b) effect of increasing the mean call length

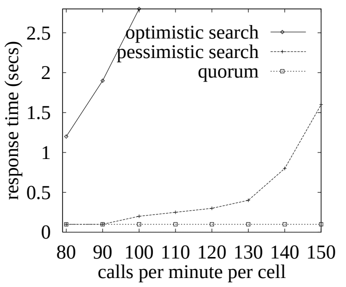
quorum algorithm, the call that requires more than 2 request attempts is dropped. These changes are made so that the data from the quorum algorithm can be effectively compared with the data from PK and CS taken from [2].

First, the comparison of the three algorithms are performed under uniform traffic distribution. Figure 10(a) shows the simulation result when the mean arrival rate varies from 35 to 60 calls per miniute per cell, and the mean call length is 180 secs. Figure 10(b) shows the result when the mean call length varies from 108 to 180 secs, and the request arrival rate is set at 60 calls per minute per cell. The response time of the quorum algorithm is significantly lower than that of PK and CS. According to [2], the dropped calls in PK and CS ranges from 0% to 21% as the call arrival rate increases. There are no dropped calls in the quorum algorithm. We do not show these results in the figures.

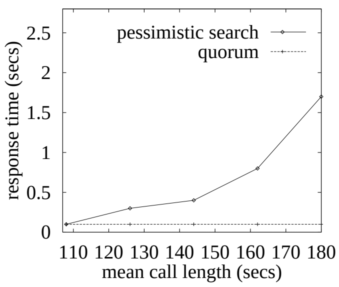
Under the non-uniform traffic distribution, the arrival rate in hot state is 5 times higher than normal state. Figure 11(a) shows the results when the arrival rate for heavily loaded cells varies from 80 to 150, and the mean length of each call is set to 3 minutes. Figure 11(b) shows the results when the mean call length varies from 108 to 150 secs, and the arrival rate is set to 150 calls per minute. The response time of PK is not shown in Figure 11(b) since it is significantly higher than the others. There are no dropped calls in the quorum algorithm. The percentage of dropped calls in PK and CS ranges from 0% to 54% as the call arrival rate increases.

9 Conclusion

We have presented an efficient distributed algorithm for channel allocation based on the update approach. Quorums are employed to reduce message complexity from the basic update algorithm. This algorithm also requires less storage overhead than the existing algorithm using the same technique. The member of our request set for each cell is fixed, not dependent upon the channel being requested. For $nR < D_{min} < (n+1)R$, only $3n$ request messages are sent to cells at most n hops away. Furthermore, the response to each request message is sent back to the requesting cell without deferring. Therefore, this algorithm is suitable for applications requiring real-time response. Our algorithm can be used with any dynamic or hybrid channel allocation strategies. As



(a)



(b)

Figure 11: The acquisition delay under non-uniform load distribution. (a) effect of increasing mean arrival rate, (b) effect of increasing the mean call length

a result, the algorithm adapts well to the dynamic change of the traffic load.

We also proposed a scheme to further reduce message complexity of our original algorithms at the expense of higher acquisition delay. The simulation is conducted to investigate how fast the number of request attempts in the modified algorithms increases as the channel demand grows in both uniform and non-uniform distributions. From the experimental results, the average number of request attempts in the quorum algorithms with acquisition messages is relatively small. Therefore, it is reasonable to conclude that the effect of livelock and starvation can be negligible. The average number of request attempts in the modified quorum algorithm exceeds the point where the modified algorithm is outperformed by the original one at the very high channel demand. Therefore, a combination of the original and modified algorithms can be used to fine-tune mobile network throughput while reducing traffic load on the underlying wired network. Finally, the simulation results show that the acquisition delay of the quorum-based algorithm is significantly lower than that of search algorithms (PK and CS).

References

[1] Manhoi Choy and Ambuj K. Singh. Efficient fault tolerant algorithms for distributed resource allocation. In *ACM Transactions on Programming Languages and Systems*, 17(4):535-559, 1995.

[2] Manhoi Choy and Ambuj K. Singh. Efficient Distributed Algorithms for Dynamic Channel Assignment. In *The Seventh IEEE International Symposium on Personal, Indoor and Mobile Rodio Communications*, pages 208-212, October 1996.

[3] Xuefeng Dong and Ten H. Lai. Distributed Dynamic Carrier Allocation in Mobile Cellular Networks: Search vs. Update. In *Proceedings of the 17th International Conference on Distributed Computing Systems*, pages 108-115, 1997.

[4] N. Garg, M. Papatriantafilou, and P. Tsigas. Distributed List Coloring: How to Dynamically Allocate Frequencies to Mobile Base Stations. In *Symposium on Parallel and Distributed Processing 1996*, pages 18-25.

[5] Chih-Lin I and Pi-Hui Chao, Local Packing - Distributed Dynamic Channel Allocation at Cellular Base Station. In *IEEE GLOBECOM 1993*, pages 293-301.

[6] Chih-Lin I and Pi-Hui Chao, Distributed Dynamic Channel Allocation Algorithms with Adjacent Channel Constraints. In *The Fifth IEEE International Symposium on Personal, Indoor and Mobile Rodio Communications*, vol. B2.3, pages 169-175, 1994.

[7] I. Katzela and M. Naghshineh. Channel Assignment Schemes for Cellular Mobile Telecommunication Systems: A Comprehensive Survey. In *IEEE Personal Communications*, pages 10-31, June 1996.

[8] L. Lamport. Time, Clocks and the Ordering of Events in a Distributed System. In *Communications of the ACM*, 21(7): 558-565, November 1978.

[9] M. Maekawa. A $\sqrt{N}$ Algorithm for Mutual Exclusion in Decentralized Systems. In *ACM Transactions on Computer Systems*, pages 145-159, May 1985.

[10] Ravi Prakash, N. G. Shivaratri, and M. Singhal. Distributed Dynamic Channel Allocation for Mobile Computing. In *Proceedings of the 14^{th} ACM Symposium on Principles of Distributed Computing*, pages 47-56, Ottawa, Canada, August 1995.

[11] G. Ricart and A. K. Agrawala. An Optimal Algorithm for Mutual Exclusion in Computer Networks. In *Communications of the ACM*, 24(1):9-17, January 1981.

A Appendix

Theorem 5.2 *For any integer $n \geq 1$, the n-hop request set satisfies $\mathcal{PN}$ if $nR < D_{min} \leq (n+1)R$*

Proof: We prove this theorem by induction on n. The base case is obtained from Lemma 5.1. Given cells c, c' such that $dist(c, c') < (n+1)R$, we need to show that $S_c \cap S_{c'} \neq \emptyset$. From induction hypothesis, for cell c' such that $dist(c, c') < nR$, we know that $S_c \cap S_{c'} \neq \emptyset$.

Let a pair (d, θ) represent the position of c' relative to c, where d is the distance between c and c', and θ is the angle between the line connecting c and c' and the horizontal axis. Hence, the position of c is $(0, 0)$. Given a cell $c' = (d, \theta)$ such that $nR \leq d < (n+1)R$, the position of cell $x \in S_c \cap S_{c'}$ is as follows:

Case (1) $\underline{d = nR}$:

(1.1) $\theta = 0$, $2\pi/3$, or $4\pi/3$
$\Rightarrow x = c'$

(1.2) $\theta = \pi/3$, π, or $5\pi/3$
$\Rightarrow x = c$

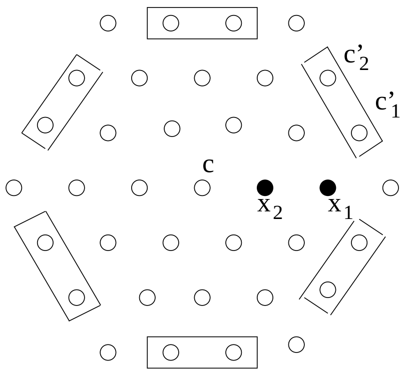


Figure 12: Positions of some $x \in S_c \cap S_{c'}$ for $D_{min} = 3R$.

Case (2) $\underline{nR < d < (n+1)R}$:

Cells in this category can be divided into 6 groups. Each group contains exactly n cells, $c'_1, \ldots, c'_n$, such that $\theta_1 < \ldots < \theta_n$, where θ_i is the angle of cell c'_i, for $i = 1, \ldots, n$.

(2.1) $0 < \theta < \pi/3$,
$\Rightarrow x_i = ((n-i+1)R, 0)$
(Figure 12 illustrates the positions of x_1 and x_2 for $n = 2$).

(2.2) $\pi/3 < \theta < 2\pi/3$,
$\Rightarrow x_i = (iR, 2\pi/3)$

(2.3) $2\pi/3 < \theta < \pi$,
$\Rightarrow x_i = ((n-i+1)R, 2\pi/3)$,

(2.4) $\pi < \theta < 4\pi/3$,
$\Rightarrow x_i = (iR, 4\pi/3)$

(2.5) $4\pi/3 < \theta < 5\pi/3$,
$\Rightarrow x_i = ((n-i+1)R, 4\pi/3)$

(2.6) $5\pi/3 < \theta < 2\pi$,
$\Rightarrow x_i = (iR, 0)$

According to the n-hop request set, it is easy to see that in each case, $x \in S_c$ and $x \in S_{c'}$.

□

Theorem 5.4 *(Safety) No two cells within each other's interference neighborhood simultaneously use the same channel.*

Proof: We show that any two cells i, j within each other's interference neighborhood cannot simultaneously use the same channel k. Without loss of any generality, assume that the timestamp of cell i's request (t_i) is smaller than that of cell j (t_j). According to the structure of the request set, there are 3 cases to consider: (1) $|S_{ij}| = 1$ and $i \in S_j$ or $j \in S_i$. (2) $|S_{ij}| = 1$ and $i \notin S_j$ and $j \notin S_i$. (3) $|S_{ij}| \geq 2$.

- Case (1): Same as the proof in Theorem 4.2.
- Case (2): Let $\phi \in S_{ij}$. If cell i's request arrives at ϕ before the request of j, then i will receive $GRNT(k)$ and j will receive $REJ(k)$ from ϕ. If cell j's request arrives at ϕ before i, then i will receive $C\_GRNT(j,k)$ and j will receive $GRNT(k)$ from ϕ. Therefore, i's request will fail because $i \notin S_j$.

  If i's request has been confirmed at ϕ before j's request arrives, then j's request will be rejected by ϕ. On the other hand, if j's request has been confirmed by ϕ before i's request arrives, then i's request will be rejected by ϕ.
- Case (3): From Property 5.3, we know that either $i \in S_j$ or $j \in S_i$.

  - If $i \in S_j$, from Step (B), j's request will be rejected by i since $t_i < t_j$. If i is using k, j's request will be rejected. If j is using k, then k must be in I_i, and i will not select k.

  - If $j \in S_i$, i's request will be granted by j. j'request will fail even though every response that j receives is either $GRNT$ or C_GRNT since condition (C3) is failed.

  If j is using k, i's request will be rejected. If i is using k, then k must be in I_j, and j will not select k.

□