Efficient Distributed Channel Allocation for Mobile Cellular Networks

Guohong Cao and Mukesh Singhal
Department of Computer and Information Science
The Ohio State University
Columbus, OH 43210
E-mail: {gcao, singhal}@cis.ohio-state.edu

Abstract

The performance of a distributed dynamic channel allocation algorithm is measured by the call blocking rate, the number of messages exchanged per channel acquisition, and the delay incurred in acquiring a channel. In general, there are two approaches to design a distributed channel allocation algorithms: *Search* and *Update*. Both of these approaches have advantages and disadvantages. The update approach has shorter acquisition delay and lower call blocking rate, but higher message complexity. On the other hand, the search approach has lower message complexity, but longer acquisition delay and higher call blocking rate. In this paper, we first propose a novel distributed acquisition algorithm, which has similar message complexity as the search approach and similar acquisition delay as the update approach. Then, we present a channel selection algorithm and integrate it into our distributed acquisition algorithm. By a rigorous analysis in terms of delay and message complexity, we show that our channel selection algorithm performs significantly better than the update approach [4] and the search approach [12]. Detailed simulation experiments are carried out in order to evaluate our proposed methodology. The performance of our algorithm is compared with those of the Geometric strategy [1], the Search approach [12], and the Update approach [4]. Simulation results show that our algorithm outperforms all other approaches in terms of call blocking probability under uniform as well as non-uniform traffic.

Key words: Distributed channel allocation, channel borrowing, cellular network.

1 Introduction

Cellular communication networks divide the geographical area into smaller regions, called cells [6]. Each cell has a *mobile service station* (MSS) and a number of *mobile hosts* (MHs). To establish a communication session (or a call), an MH sends a request to the MSS in its cell. The session is supported if a wireless channel can be allocated for the communication between the MH and the MSS. Since the frequency spectrum is limited, the frequency channels must be reused as much as possible in order to support the increasing demand for wireless communication. However, two different cells cannot use the same channel if their geographic distance is less than a threshold, called the *minimum channel reuse distance* (D_{min}) [1, 11]; otherwise, the communication sessions will interfere with each other.

A channel is *available* for a cell if its use in the cell does not interfere with others. When a cell needs a channel, it acquires one available channel using a channel allocation algorithm. A channel allocation algorithm consists of two parts: a channel acquisition algorithm and a channel selection algorithm. The channel acquisition algorithm is responsible for collecting information from other cells and making sure that two cells within D_{min} do not use the same channel. The channel selection algorithm is used to choose a channel from a large number of available channels in order to achieve better channel reuse. The performance of a channel acquisition algorithm is measured by the message complexity and the acquisition delay. The message complexity is measured in terms of the number of messages exchanged per channel acquisition. The acquisition delay is the time required for a MSS to allocate a channel. The performance of the channel selection algorithm is measured by the *call blocking rate*. A call is blocked if there is no channel available for use when the call is being set up or when it is being handed over to another cell due to a host mobility.

Channel selection algorithms have been extensively studied [1, 5, 6, 7, 8, 11, 14]. However, most of them depend on a *Mobile Switching Center (MSC)* to accomplish channel acquisition, which are referred to as the *centralized* channel acquisition algorithms. More specifically, each cell notifies the MSC when it acquires or releases a channel so that the MSC knows which channels are available in each cell at any time and assigns channels to cells accordingly. Obviously, the centralized approach has poor reliability and scalability.

Recently, distributed channel acquisition algorithms [4, 12] have received considerable attention because of their high reliability and scalability. In this approach, an MSS communicates with other MSSs directly to find the available channels and to make sure that assigning a channel does not cause interference with other cells. In general, there are two approaches to design distributed channel acquisition algorithms: *Search* [12] and *Update* [4]. In the search approach, when a cell needs a channel, it searches all neighboring cells to find the set of currently available channels, and then picks one according to the channel selection strategy. In the update approach, a cell maintains information about available channels. When a cell needs a channel, it selects an available channel according to the underlying channel selection strategy and consults the neighboring cells whether it can acquire the selected channel. Also, a cell informs its neighbors each time it acquires or releases a channel, so that each cell has up-to-date information on the available channels. Both of these approaches have advantages and disadvantages. The update approach has shorter acquisition delay and good channel reuse, but it has higher message complexity. On the other hand, the search approach has lower message complexity, but it has longer acquisition delay and poor channel reuse.

In this paper, we first propose a novel distributed acquisition algorithm, which has similar message complexity as the search approach and similar acquisition delay as the update approach. Then, we present a channel selection algorithm and integrate it into our distributed acquisition algorithm. Based on a rigorous analysis in terms of delay and message complexity, we show that our algorithm performs significantly better than the update approach and the search approach. Detailed simulation experiments are carried out to evaluate our proposed methodology. The performance of our algorithm is compared with those of the Geometric strategy [1], the Search approach [12], and the Update approach [4]. Simulation results show that our algorithm outperforms all other approaches in terms of call blocking probability under uniform and non-uniform traffics.

The rest of the paper is organized as follows. Section 2 presents the system model. In Section 3, we propose a distributed channel acquisition algorithm and give a correctness proof. Section 4 presents a channel selection algorithm and integrates it into our channel acquisition algorithm, and then we compare the complete channel allocation algorithm with the search and the update approach in terms of message complexity and acquisition delay. In Section 5, we present our simulation results. Section 6 concludes the paper.

2 System Model

The geographical area is divided into hexagonal cells in a mobile cellular network [6]. Each cell is supported by a MSS in the center (we use cell and MSS interchangeably). The MSSs are connected to each other by a static wired network, which provides reliable $FIFO$ communications. To establish a communication session (or a call), an MH sends a request to the MSS in its cell. The session can be supported if a wireless channel can be allocated for communication between the MH and the MSS. Two cells can use the same channel if the geographic distance between them is at least as big as a threshold D_{min}; otherwise, their communication sessions interfere with each other, which is referred to as *channel interference.*

Definition 1 Given a cell C_i, the set of *interference neighbors* of C_i, denoted by IN_i, is:

$$IN_i = \{C_j | distance(C_i, C_j) < D_{min}\}.$$

As shown in Figure 1, R is the cell radius, D_{min} is the minimum channel reuse distance. If a fourth-power law attenuation is assumed [1, 10], the signal to interference ratio is given by $[S/I]_{min}$ $= [(D_{min}/R) - 1]^4/6$. With $D_{min} = 3\sqrt{3}R$, $[S/I]_{min} \approx 17dB$, which is a reasonable value in practice.

Resource Planning Model

Most of channel selection strategies require a *priori* information about channel status in order to achieve better channel reuse. For instance, in the channel borrowing strategies [5, 7, 11], each cell is allocated a set of "nominal" channels before hand; in the geometric strategy [1], each cell must know its "first-choice" channels prior to any channel acquisition. We call the process of assigning

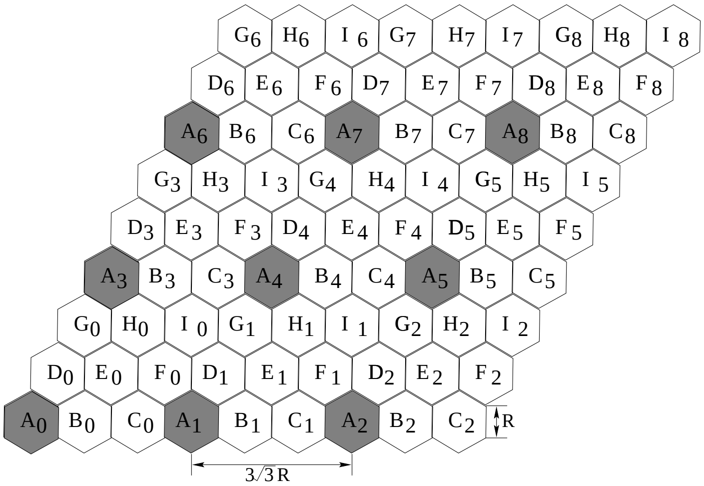

Figure 1: An optimal partition

special status to channels as the *resource planning* [3, 4].

Resource Planning

1. Partition the set of all cells into a number of disjoint subsets, $G_0, G_1, \cdots, G_{k-1}$, such that any two cells in the same subset are apart by at least a distance of D_{min}. Accordingly, partition the set of all channels into k disjoint subsets: $P_0, P_1, \cdots, P_{k-1}$.
2. The channels in $P_i (i = 0, 1, \cdots, k-1)$ are *primary channels* of cells in G_i, and *secondary channels* of cells in $G_j (j \neq i)$.
3. A cell requests a secondary channel only when no primary channels are available.

For convenience, we say that a cell C_i is a *primary (secondary)* cell of a channel r if and only if r is a primary (secondary) channel of C_i. Thus, the cells in G_i are primary cells of the channels in P_i and secondary cells of the channels in P_j $(j \neq i)$.

Definition 2 For a cell $C_i \notin G_p$ and a channel $r \in P_p$, the *interference primary cells* of r relative to C_i, denoted by $IP_i(r)$, are the cells which are primary cells of r and interference neighbors of C_i; i.e., $IP_i(r) = G_p \cap IN_i$. $IP_i(r)$ is also referred to as an *interference partition subset* of C_i.

In order to achieve better channel reuse, each subset G_i should contain as many cells as possible. Then, the k should be as small as possible. How to partition the cells is orthogonal to our discussion, but we require that the partitions satisfies the following property according to the resource planning model, which is obvious:

Property 1: $\forall C_i, C_j \in G_p : distance(C_i, C_j) \geq D_{min}$

Figure 1 shows one partition, which divides the cells into nine subsets $G_A, G_B, \cdots, G_I$. Cells in $G_A = \{C_{A_i} | 0 \leq i \leq 8\}$ can use the same channel without interference. Since the distance between any two nearest cells in a subset is exactly D_{min}, it is an optimal partition in the sense that each channel is maximally reused by its neighbors.

Handoff and intra-Handoff

A mobile host may cross the boundary between two cells while being active. When this occurs, the necessary state information must be transferred from its previous MSS to the MSS in the new cell. This process is known as *handoff* (or *inter-handoff*) [10]. During a handoff, an MH releases its current channel to its previous MSS and is assigned a new channel by the new MSS.

To achieve better channel reuse, *intra-handoff* (or a channel switch) may be necessary [2, 4]. In an intra-handoff operation, an MH releases its current channel and is assigned a new channel within the same cell. The motivation behind intra-handoff can be understood by an example. In Figure 1, suppose cell C_{F_1} borrows a channel $r1$ from A_1 and assigns it to a mobile host MH_f. Then, cells C_{A_1}, C_{A_2}, C_{A_4}, and C_{A_5} cannot use channel $r1$ due to interference. If a call in C_{F_1} terminates and a primary channel $r2$ is released, an intra-handoff from $r1$ to $r2$ by MH_f improves channel reuse, since $r1$ can be reused by four other cells C_{A_1}, C_{A_2}, C_{A_4}, and C_{A_5}. A drawback of intra-handoff is of course the overhead. Fortunately, most of channel selection strategies do not demand many intra-handoffs [2, 4]. Thus, intra-handoff may be necessary for better channel reuse.

3 A Distributed Channel Acquisition Algorithm

In this section, we investigate the fundamental difference between the search and the update approaches. Then, we present our distributed channel acquisition algorithm and provide a correctness proof.

3.1 Search vs. Update

Both the search and the update approaches timestamp control messages using Lamport's logical clocks [9] to determine the priority of requests.

3.1.1 The Search Approach

In the search approach [12], when a cell C_i needs a channel, it changes to *search mode* and sends *request* messages to each cell in IN_i. When a cell C_j receives a *request* from C_i, if C_j is not in search mode or C_j is in search mode but C_j's request has higher timestamp than C_i's, C_j sends a *reply* message to C_i which contains information about its used channels; otherwise, C_j defers the *reply* (similar to [13]). After C_i has received all the *reply* messages from each cell in IN_i, it computes the available channels and picks one, say r, from them. C_i sends *confirm* messages to the lenders of r. If all lenders reply *agree*, C_i can use r; otherwise, C_i picks another available channel and repeats the process.

3.1.2 The Update Approach

In the update approach [4], a cell maintains information about the available channels. When a cell C_i needs a channel, it picks an available channel r according to the underlying channel selection strategy and then sends a *request* message to each cell in IN_i. A cell that receives a *request*, replies with a *reject* if either it is using r or it is also requesting for r with a smaller timestamp; otherwise, it replies with an *agree*. If C_i has received *agree* messages from all the cells in IN_i, C_i notifies them that it has successfully acquired channel r; otherwise, C_i picks another available channel and repeats the process. When C_i finishes the use of the borrowed channel, it sends a *release* to each cell in IN_i.

3.1.3 A Comparison

In the search approach, a cell communicates with its interference neighbors only when it needs to borrow a channel. However, in the update approach, a cell keeps communicating with its interference neighbors in order to get the up-to-date information. Clearly, the update approach is likely to have significantly higher message complexity than the search approach does.

In the search approach, a cell needs to confirm a selected channel, which doubles the acquisition delay. Many good channel selection strategies rely on a cost function to determine which channel to borrow and which channel to release. For example, when a newly available channel has a higher priority than a used channel does, an intra-handoff is necessary to achieve better channel reuse. In the update approach, a cell maintains all the necessary information about its interference neighbors in order to use the cost function. Thus, this approach can support many channel selection strategies. However, in the search approach, a cell collects neighbor information only after a *search*. But the collected information maybe outdated when the call releases a channel. Therefore, many good channel selection strategies cannot be supported in the search approach. Moreover, the search approach [12] locks the borrowed channel during channel borrowing, which also reduces channel reuse.

In the following, we propose a channel acquisition algorithm which reduces the acquisition delay and does not lock the borrowed channel (A channel selection algorithm to achieve better channel reuse is discussed in Section 4.2).

3.2 Our Distributed Channel Acquisition Algorithm

3.2.1 Reducing the Acquisition Delay

In the search approach, a cell has to $confirm$ a selected channel with the lenders (interference primary cells) since a lender may assign that channel to a new call immediately after it sent a $reply$. One way to avoid $confirm$ is to let the channel lenders wait until they know which channel the borrower has selected. However, this requires all interference neighbors to lock their channels for $2 * T$ (T is one way communication delay), which may not be desirable most of the time. Our solution to this problem is as follows. When a cell receives a $request$, it marks some channels as $reserved$ channels and then sends its channel information to the borrower. The borrower selects a channel using its underlying channel selection algorithm. If the selected channel is not a $reserved$ channel, it can use the selected channel without confirming with the lenders. Otherwise, it needs to $confirm$ with the lenders similar to the search approach. In both situations, a cell sends $finish$ (or $transfer$) messages to its interference neighbors before it starts using the borrowed channel. For any interference neighbor, if a call arrives during the channel borrowing process, it assigns a reserved channel to the call. If a call arrives after a cell has used all its reserved channels, the cell cannot assign any other available channels to this call until it receives the $finish$ (or $transfer$) messages.

3.2.2 Notations

The following notations are used in our channel acquisition algorithm.

- IN_i, $IP_i(r)$: defined before.
- S: the set of all the channels in the system.
- P_i: the set of primary channels assigned to C_i.
- U_i: the set of used channels at C_i.
- A_i: the set of currently known available channels at C_i.
- $reserved_i$: the set of reserved channels at C_i. C_i has at most N_i reserved channels, where N_i is a system tuning parameter (see Section 5.1).
- $pick(A)$: pick a channel r from a set of available channels A using the underlying channel selection algorithm.
- $send_i$: the set of cells to which C_i has sent a $reply$ message but has not received the $finish$ (or $transfer$) message.
- $I_i(r)$: the set of cells to which C_i has sent an $agree(r)$. If $I_i(r) \neq \phi$, r is an interference channel of C_i. Then, C_i cannot use r, but it can lend it to other cells.

3.2.3 The Channel Acquisition Algorithm

In the distributed channel acquisition algorithm, if a cell has an available primary channel r, it uses r immediately unless an interference neighbor is in the search mode. If C_i does not have any available primary channel, it searches all neighboring cells to find the set of available channels and picks one from them. When C_i borrows a channel r, the interference primary cells of r relative to C_i cannot use r until C_i returns r.

A formal description of the algorithm is given in Figure 2. If no channel is selected from the reserved channels, five types of messages are exchanged among MSSs to borrow or return a channel: *request*, *reply*, *finish* (or $transfer$), and *release*. Otherwise, two additional messages are needed: *confirm* and *abort*.

In the channel acquisition algorithm, several call requests may arrive when a cell is in the search mode. Then the cell can just pick more channels and assign them to these call requests. For simplicity, this is not explicitly presented in the algorithm.

3.3 Correctness Proofs

Theorem 1 *The distributed channel acquisition algorithm ensures that a cell and its interference neighbors do not use the same channel concurrently.*

Proof. Assume the contrary that two cells C_i and C_j ($C_i \in IN_j$) are using the same channel r. Since the distance between two primary cells is at least D_{min} (Property 1), C_i and C_j cannot both be primary cells of r. Hence, at least one of them is a secondary cell of r.

Case 1: C_i is a primary cell of r and C_j is a secondary cell of r. Then $C_i \in IP_j(r)$. When C_i receives its own call request, depends on the condition of $reserved_i$, there are three possibilities.

Case 1.1: $send_i = \phi$. C_i uses r to support the call request, and adds r to U_i (Step A.1). When C_j receives C_i's $reply(U_i, reserved_i)$, $r \in U_i \Longrightarrow r \notin A_j$ according to Step C.1. Then C_j cannot acquire r.

Case 1.2: $send_i \neq \phi \wedge r \in reserved_i$. C_i uses r to support the call request, and adds r to U_i (Step A.2). In order to use r, C_j sends $confirm(r)$ to C_i, but C_i rejects this $confirm(r)$.

Case 1.3: $send_i \neq \phi \wedge r \notin reserved_i$. There are two possibilities:

Case 1.3.1: $C_j \in send_i$. C_i waits until C_j acquires r (Step A.2).

Case 1.3.1: $C_j \notin send_i$. If C_j's request arrives before $send_i = \phi$, it is similar to Case 1.3.1; otherwise, it is similar to Case 1.1.

Case 2: C_j is a primary cell of r and C_i is a secondary cell of r. Similar to Case 1.

Initialization

For any cell C_i, A_i and P_i are set to be the set of primary channels assigned to C_i. U_i, $reserved_i$, and $send_i$ are set to empty. For any channel $r \in P_i$, $I_i(r)$ is set to empty.

(A) When a cell C_i needs a channel to support a call request, if $A_i = P_i - U_i - \{r \mid I_i(r) \neq \phi \wedge r \in P_i\}$ is empty, it sets its search mode, and sends a *request* message to each cell $C_j \in IN_i$; otherwise, it does the follows:

(A.1) $send_i = \phi$: let $r = pick(A_i)$, it adds r to U_i, and then uses r to support the call. When the call is finished, it deletes r from U_i.

(A.2) $send_i \neq \phi$: if $reserved_i$ is not empty, let $r = pick(reserved_i)$, C_i adds r to U_i, deletes r from $reserved_i$, and then uses r to support the call; otherwise, it defers the call request until $send_i$ is empty.

(B) When a cell C_i receives a *request* message from C_j, if it is also in the search mode and its *request* has smaller timestamp than C_j's *request*, it defers its response until it resets its search mode; otherwise, it does the follows:

(B.1) $send_i = \phi$: C_i picks N_i channels and assigns them to $reserved_i$, then it adds C_j to its $send_i$ and sends $reply(U_i, reserved_i)$ to C_j.

(B.2) $send_i \neq \phi$: C_i adds C_j to its $send_i$ and sends $reply(U_i, reserved_i)$ to C_j.

(C) When a cell C_i receives all *reply* messages from cells in IN_i:

(C.1) It determines the currently available channels as follows:
$$A_i = S - U_i - \{r \mid I_i(r) \neq \phi \wedge r \in P_i\} - \bigcup_{k \in IN_i} U_k$$

(C.2) If A_i is empty, C_i drops the call, resets its search mode, and processes all the deferred *reply* messages. If A_i is not empty, let $r = pick(A_i)$, if there is any cell C_j such that $C_j \in IP_i(r) \wedge r \in reserved_j$, C_i sends a $confirm(r)$ to C_j; otherwise, the request is successful.

(C.3) In case of success, C_i resets its search mode, adds r to U_i, sends $transfer(r)$ to every cell in $IP_i(r)$, sends *finish* to every cell in $IN_i - IP_i(r)$, and processes all the deferred *reply* messages. When C_i finishes the call, it sends $release(r)$ to every cell in $IP_i(r)$.

(D) When a cell C_i receives a $confirm(r)$ from C_j, if it is not using r, it deletes r from $reserved_i$ and responses with an *agree*; otherwise, it responses with a *reject*.

(E) If C_i receives an *agree* message from each cell to which it has sent a $confirm(r)$, the request is successful, go to Step C.3; otherwise, C_i deletes r from A_i, sends an $abort(r)$ to every cell from which it has received an *agree*, and then goes to Step C.2.

(F) When a cell C_i receives an $abort(r)$, $release(r)$, *finish*, or $transfer(r)$, it does the following:

$abort(r)$: C_i adds r to $reserved_i$.

$release(r)$: C_i deletes C_j from $I_i(r)$.

finish: C_i deletes C_j from $send_i$. If $send_i$ is empty, it processes all deferred call requests.

$transfer(r)$: C_i adds C_j to $I_i(r)$ and deletes C_j from $send_i$. If $send_i$ is empty, it processes all deferred call requests.

Figure 2: The Distributed Channel Acquisition Algorithm

Case 3: both C_i and C_j are secondary cells of r. In order to borrow channel r, C_i and C_j must have received each other's *reply* message. Without loss of generality, we assume that C_i's *request* has smaller timestamp than C_j's *request*. Then, C_j receives C_i's *reply* after C_i has borrowed channel r and added r to U_i. When C_j receives C_i's $reply(U_i, reserved_i)$, $r \in U_i \Longrightarrow r \notin A_j$ according to Step C.1. Then, C_j cannot acquire r. □

Theorem 2 *The distributed channel acquisition algorithm is deadlock free.*

Proof. New channel requests originating concurrently in different cells get totally ordered by their timestamps. An MSS in search mode sends *reply* messages to all *requests* with a lower timestamp and defers others. As the same ordering of channel requests is seen by all the MSSs, there is no circular deferring of *replys* among the MSSs.

Since the communication link is reliable, the MSS whose *request* has the highest priority can always receive all *reply* messages from its interference neighbors and determine whether to $confirm$ the selected channel. An MSS receiving a $confirm$ responds immediately either by an *agree* or a $reject$ message. Then, the MSS whose request has the highest priority can always decide whether it can successfully borrow a channel or not. Then, it processes the deferred reply and sends $finish$ or $transfer$ messages. The MSS deferring its own call request can always receive $finish$ or $transfer$ message, empty $send_i$, and then process the deferred call request. □

4 A Complete Channel Allocation Algorithm

In this section, we provide a channel selection algorithm and integrate it into our channel acquisition algorithm. Then, we give a performance analysis of our channel allocation algorithm and compare it with the search and the update approaches.

4.1 The Channel Selection Algorithm

Similar to the Geometric strategy [1] and the channel selection algorithm in the update approach [4], our channel selection algorithm makes use of the optimal resource planning model defined in Section 2. The primary channels for each cell are prioritized. During a channel acquisition, a cell acquires the available primary channel that has the highest priority. If none of the primary channels is available, the cell borrows a channel from its neighbors according to some priority assignment approach. When a cell acquires a channel, it always acquires the channel with the highest priority. When a cell releases a channel, it always releases the channel with the lowest priority. If a newly available channel has a higher priority than some used channel, an intra-handoff is performed.

Our algorithm is different from the Geometric and the update approaches when assigning priority to secondary channels. Similar to [15], in our algorithm, a cell borrows the channel that has the lowest priority from the "richest" interference neighbors; i.e., the cell with the most available primary channels. The motivation behind this is to reduce the chance that the lender might soon use up its primary channels and have to acquire a secondary channel. In the following, we formally define the channel priority used in our channel selection algorithm.

Let the cells be partitioned into k disjoint optimal reuse patterns, $G_0, G_1, \cdots, G_{k-1}$, as defined in Section 2. Without loss of generality, we assume that there are a total of $k*n$ channels numbered $0, 1, \cdots, k*n-1$ which are evenly divided into k subsets: $P_0, P_1, \cdots, P_{k-1}$ (This assumption is not essential and is made only for ease of presentation). Let $P_l = \{l*n, l*n+1, \cdots, (l+1)*n-1\}$, $l = 0, 1, \cdots, k-1$.

Definition 3 Given a cell $C_i \notin G_p$, the "richness" of any cell in G_p relative to C_i, denoted by $\delta_i(G_p)$ is measured as the minimum number of primary channels which are available in the interference primary cells of P_p:

$$\delta_i(G_p) = min\{\mid A_j \mid : C_j \in G_p \cap IN_i\}$$

Based on Definition 3, the priority of a channel $l*n+j$ in cell C_i is defined as follows:

$$\mathcal{P}_i(l*n+j) = \begin{cases} m-j & \text{if } l = i \\ j + \delta_i(G_l)*o & \text{if } l \neq i \end{cases} \tag{1}$$

where $0 \leq i < k$, $0 \leq l < k$, $0 \leq j < n$, and $m \gg o \geq n$, e.g., $m = 3*n*n$, $o = n$.

From Equation (1), the primary channels in a cell have the highest priority since m is a significant large number. For secondary channels, the channel from the "richest" cell has the highest priority since o is a factor larger than or equal to n. If two channels have the same "richness", the channel with higher number has higher priority.

4.2 The Complete Channel Allocation Algorithm

In order to combine the channel selection algorithm with our channel borrowing algorithm, we modify Step A.1 and A.2 of our channel acquisition algorithm as follows: when a cell acquires or releases a primary channel, it notifies all cells which have borrowed channels from it. More formally, when a cell C_i acquires or releases a primary channel $r \in P_i$, it sends *update*(r) to each cell in $\{C_j \mid r' \in P_i \wedge C_j \in I_i(r')\}$. When a cell C_i receives an *update*(r) from C_j, it updates the local information about C_j, and then it keeps the up-to-date information for calculating the "richness" of its interference neighbors.

Note that our channel acquisition algorithm is independent of the channel selection algorithm being used. We can use the same channel selection algorithm as that of the update approach or any other newly developed channel selection algorithm. The use of the proposed channel selection algorithm is just to show how to support a complex channel selection algorithm which makes use of up-to-date information to achieve better channel reuse.

The key difference between the update approach and our algorithm is that our algorithm only notifies those cells to which it has lent channels in stead of to all its interference neighbors. Since the number of borrowers is very small compared to the number of interference neighbors, our algorithm significantly reduces the update notification message complexity compared to the update approach.

4.3 Performance Analysis and Comparison

We analyze the performance of our channel allocation algorithm and compare it to the search [12] and the update approaches [4]. Let n_p denote the number of interference primary neighbors of a cell.

The number of messages per primary channel acquisition and the primary channel acquisition delay are both 0. If there is no need to confirm with the lenders, the average secondary channel acquisition delay is $2 * T$ and the number of messages per secondary channel acquisition is $3n + n_p$, which includes n number of $request$ and $reply$ messages, $(n - n_p)$ number of $finish$, and n_p number of $transfer$ and $release$ messages. If a cell confirms m $(m \geq 1)$ times before it acquires a channel, the number of messages per secondary channel acquisition is $3n + n_p + 2 * m * n_p + (m - 1) * n_p$, where $2 * m * n_p$ indicates m rounds of maximum n_p number of $confirm$ and $agree$ (or $reject$) messages, and $(m - 1) * n_p$ indicates the maximum number of $abort$ messages for the $m - 1$ times failed $confirm$.

Table 1: **Comparison of the update, the search, and our approach**

	Messages Complexity	Acquisition Delay
The Search Approach	$\alpha * (2 * n + 3 * n_p * (1 + m'))$	$2 * T * (2 + m') + T'_d$
The Update Approach	$2 * n + \alpha * (3 * n_p * m'' + 2 * n_p)$	$2 * T * (1 + m'') + T''_d$
Our Approach	$\leq \alpha * (3 * n + 3 * n_p * m''' + n_p) + n_u$	$2 * T * (1 + m''') + T'''_d$

Let α denote the percentage of secondary channel acquisition, n_u denote the number of $update$ messages needed in our algorithm, $m(m', m'', m''')$ denote conflict rates, and $T_d(T'_d, T''_d, T'''_d)$ denote the extra deferred delay due to a conflict. Table 1 lists the average number of message per channel acquisition and the average secondary channel acquisition delay in the search approach, the update approach, and our approach.

In a typical cellular network model with $D_{min} = 3\sqrt{3}R$, we have $n = 30$ and $n_p = 3$ or 4. Normally (according to the simulation), m and T_d are both very small compared to T; then, from Table 1, our algorithm almost cuts the acquisition delay to half compared to the search approach.

When the channel request load is low, most of time, it is not necessary for a cell to borrow channels from others; thus, both α and n_u are near 0 under low channel request load. When the channel request load increases, more cells run out of primary channels and have to make more secondary channel acquisitions, then the value of α and n_u increases. Normally (according to the simulation results), even when the channel request load increases to 100%, α is still less than 0.3 and n_u is far smaller than n (Note that, if we use the Geometric strategy or the update approach as the underlying channel selection algorithm, n_u is 0). Thus, our approach significantly reduces the message complexity compared to the update approach whose message complexity is always larger than $2 * n$ (see Table 1).

Table 2: **Simulation Parameters for uniform traffic distribution**

Mean arrival rate in a cell	λ
Mean inter-handoff rate in a normal cell	1/60s
Mean service time per communication session	180s

5 Simulation Results

We studied the performance of the proposed channel allocation algorithm, the Search approach, the Update approach, and the Geometric strategy using extensive simulations. The performance of each strategy is simulated under both uniform and non-uniform traffic distributions.

5.1 Simulation Parameters

The simulated cellular network is a wrapped-around layout with 15*15 cells. The total number of channels in the system is 396. With $D_{min} = 3\sqrt{3}R$, each cell is assigned 396/9=44 channels. Since a single bit is enough to represent whether the channel is used or not, the control message in our algorithm is very small compared to the header. Considering an MSS may not respond immediately to incoming messages, we assume the average one-way communication delay between two MSSs to be 2 milliseconds, which covers the transmission delay, the propagation delay, and the message processing time.

Under uniform traffic distribution (shown in Table 2), traffic in each cell is characterized by the mean inter-arrival time, mean service time, and mean inter-handoff time, all assumed to be negative exponentially distributed.

Under non-uniform traffic distribution, a cell can be in one of two states: *hot state* or *normal state*. As shown in Table 3, a cell spends most of its time in the normal state. A cell in the normal state is characterized by low arrival rate and high inter-handoff rate. On the contrary, a cell in hot state is characterized by high arrival rate and low inter-handoff rate to picture more arriving new users and prevailing stationary users. Also, the state change rate is assumed to be negative exponentially distributed.

We assume N_i in our algorithm is 1. If N_i is 0, all the interference neighbors of a borrower have to lock their channels for $2 * T$ time limit, which is not affordable. With $N_i = 1$, when a lender receives a new call from its own cell during its locking period (from it sends out a $reply$ to receive a $finish$ or $transfer$), it can assign the $reserved$ channel to the new call, but it needs to wait if it receives another new call during the locking period. Under 100% channel request load, our simulation shows that the probability for any interference neighbor of a borrower receiving two calls during the locking period is as low as 0.00003, which is negligible. With $N_i > 1$, the probability of receiving two calls during the locking period can be further reduced. However, the probability for the borrower to select a $reserved$ channel becomes larger. When a cell selects a $reserved$ channel, it needs an extra round of $confirm$ messages and doubles the acquisition delay. For example, with $N_i = 2$, the extra delay due to $confirm$ at the borrower side is nearly doubled compared to $N_i = 1$ from our simulation. Since 0.00003 is already a very small number, further reducing the probability of waiting at the lender side by increasing N_i cannot compensate for the increased delay at the borrower side. Thus, we only consider $N_i = 1$ in the following simulation.

Table 3: **Simulation parameters for non-uniform traffic distribution**

Mean arrival rate in a normal cell	λ
Mean arrival rate in a hot cell	3λ
Mean inter-handoff rate in a normal cell	1/60s
Mean inter-handoff rate in a hot cell	1/180
Mean rate of change from normal state to hot state	1/1800s
Mean rate of change from hot state to normal state	1/180s
Mean service time per communication session	180s

5.2 Simulation Results

For each arrival rate, the mean value of a measured parameter is obtained by collecting a large number of samples such that the confidence interval is reasonably small. In most cases, the 95% confidence interval for the measured data is less than 10% of the sample mean.

Our simulation shows that m' and m'' (Table 1) are both below 0.01%. T_d', T_d'' and T_d''' are all below 1%. Thus, the effect of these parameters is negligible, and they are not considered in analyzing the message overhead and acquisition delay for the sake of simplicity.

5.2.1 Message Complexity Per Channel Acquisition

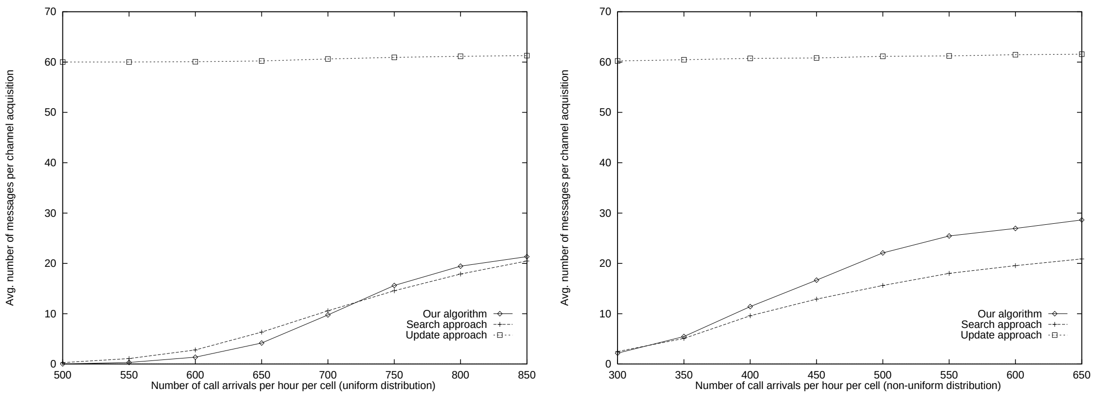

Figure 3: Comparisons of message complexity

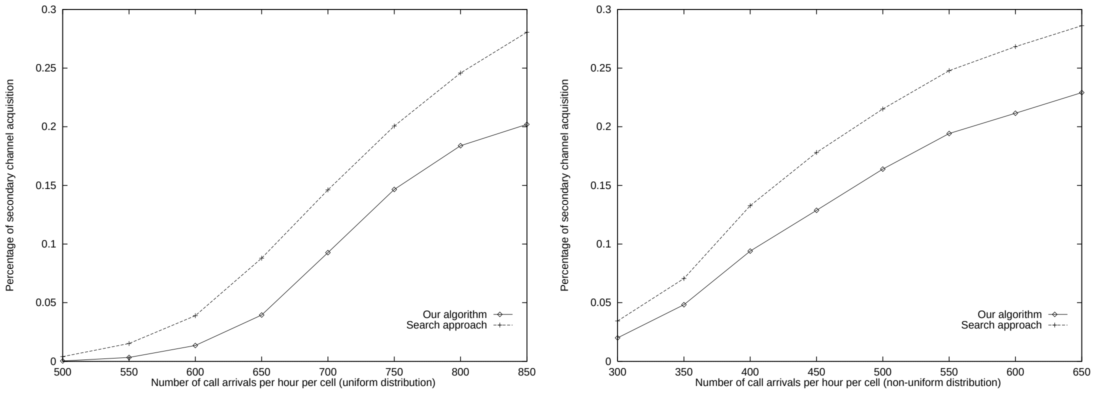

Figure 4: Percentage of secondary channel acquisition

As shown in Figure 3, the number of messages per channel acquisition in the update approach is never lower than $2 * n = 60$. In the search approach and the proposed algorithm, the message complexity increases from near 0 to about 20 as the channel request load increases. As analyzed in Section 4.3, the message complexity in the search approach and the proposed algorithm is decided by the number of secondary channel acquisition. Under low channel request load, most of the call requests can be satisfied by the primary channel acquisition. As channel request load increases, more cells run out of primary channels and have to make more secondary channel acquisitions.

From the analysis in Section 4.3, our algorithm has higher message complexity $(3 * n)$ than the search approach $(2 * n)$, but Figure 3 shows that sometimes our algorithm has lower message complexity than the search approach. This can be explained by the fact that both algorithms have different secondary channel acquisition percentage. From Figure 4, we can see that the search approach has higher secondary channel acquisition rate than our algorithm does, since the search approach does not consider channel reuse; that is, the search approach does not consider resource planning. Thus, a cell is more likely to run out of primary channels in the search approach than that in our algorithm. Also, in our algorithm, a cell only borrows a channel from the "richest" interference neighbors, which reduces the chance that the lender might soon use up its primary channels and have to acquire a secondary channel, but in the search approach, a cell just randomly borrows a channel from its neighbors.

Note that the message complexity of our algorithm also includes update notification messages, which is shown in Figure 5 (Note that, if we use the Geometric strategy or the update approach as the underlying channel selection algorithm, there is no update notification message overhead). The update notification message complexity is determined by the number of secondary channels and other factors, such as how long a borrower keeps the borrowed channel. In general, a borrower keeps the borrowed channel longer under non-uniform traffic than that under uniform traffic. Thus, there are more update notification messages under non-uniform traffic than those under uniform traffic. An interesting fact is that the number of update notification messages drops after some point (750 under uniform traffic). This can be explained by the fact that the call blocking rate increases faster than the secondary channel acquisition percentage after that point.

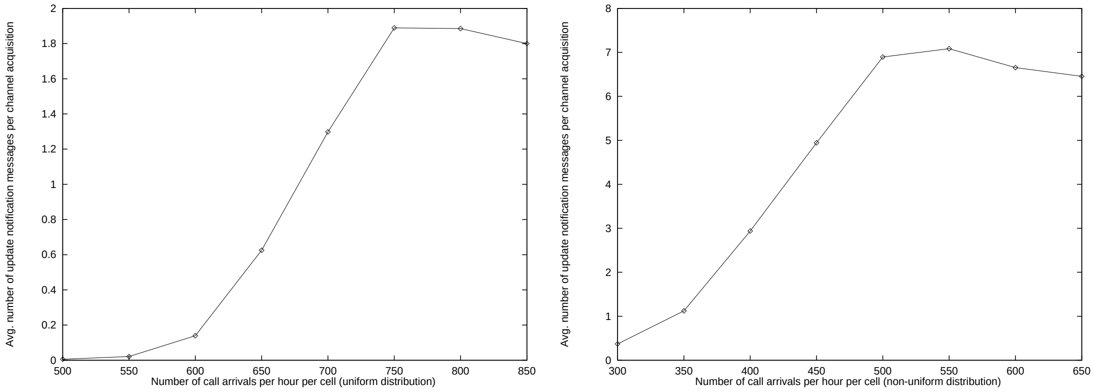

Figure 5: The number of update notification messages

5.2.2 Acquisition Delay per Channel Transfer

As shown in Figure 6, the search approach has the highest acquisition delay. Both our algorithm and the update approach almost reduce the acquisition delay by half as compared to the search approach. In our algorithm, a cell may select a *reserved* channel, which results in an extra round of *confirm* messages and doubles the acquisition delay. From Figure 6, the chance of selecting a reserved channel is less than 8%. If there is a time constraint, our algorithm can be modified so that another channel is selected if a reserved channel is chosen. However, this may increase the call blocking rate.

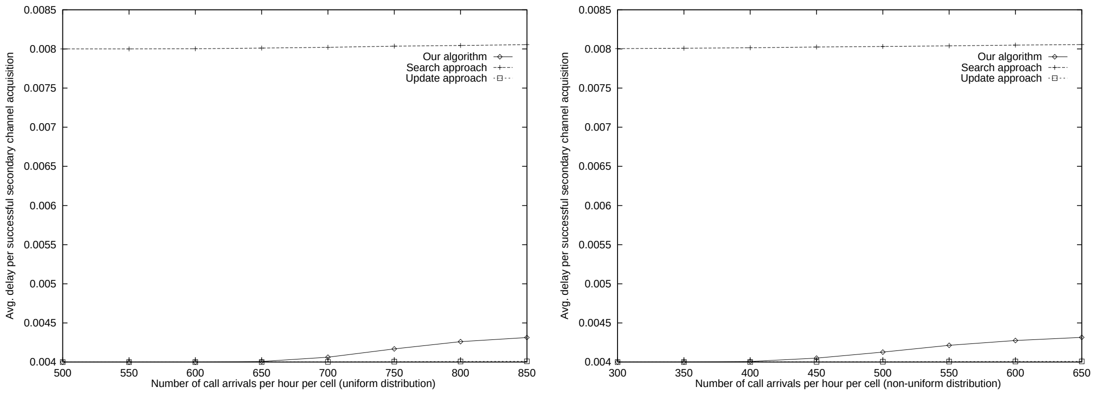

Figure 6: Comparisons of acquisition delay

5.2.3 A Comparison of Call Blocking Rate

The blocking rate of our algorithm is compared with the Geometric strategy, the Search approach, and the Update approach. Since the geometric strategy, the update approach, and our algorithm are all based on the optimal resource planning model, the call blocking rate in these three approaches does not have too much difference, with our algorithm slightly outperforming the other two. This can be explained as follows. In our algorithm, a cell only borrows a channel from the "richest" interference neighbors, which reduces the chance that the lender might soon use up its primary channels and have to acquire a secondary channel. In the update approach, the "richness" is partially considered, while the "richness" is not considered in the geometric strategy. These explains why our algorithm outperforms the update approach which outperforms the geometric strategy.

Compared to the search approach, our algorithm significantly reduces the call blocking rate. This is due to the fact that the search approach is a simple channel borrowing approach without considering any channel reuse. Moreover, the search approach locks the borrowed channel during channel borrowing, which also reduces channel reuse.

We did not consider the effect of intra-handoff since almost all advanced channel selection algorithms use intra-handoff to reduce the call blocking rate. The geometric strategy, the update approach, and our algorithm all use intra-handoff to achieve better channel reuse. The search approach does not have intra-handoff, but its call blocking rate is significantly higher than the other three.

6 Conclusions

Distributed channel allocation algorithms have received considerable attention because of their high reliability and scalability. However, these algorithms are based either on the update approach which

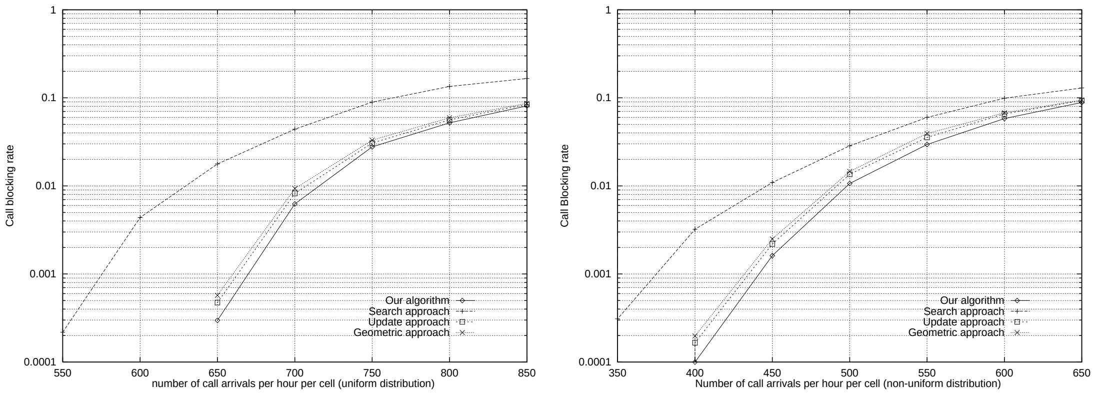

Figure 7: Comparisons of call blocking rate

has high message complexity or on the search approach which has long acquisition delay and high call blocking rate. In this paper, we proposed an efficient distributed channel allocation algorithm that keeps the advantages and avoids the disadvantages of both approaches. Compared to the search approach, our algorithm significantly reduces the call blocking rate and cuts the acquisition delay by almost half. Compared to the update approach, our algorithm avoids the message overhead under low channel request load. Even under high channel request load, our algorithm reduces the message overhead by at least a factor of two.

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 Transactions on Vehicular Technology*, pages 14–23, February 1995.

[2] R. Beck and H. Panzer. "Strategies for Handover and Dynamic Channel Allocation in Micro-Cellular Mobile Radio Systems". *IEEE Vehicular Tech. Conf.*, May 1989.

[3] X. Dong and T.H. Lai. "An Efficient Priority-based Dynamic Channel Allocation for Mobile Cellular Networks". *INFOCOM*, 1997.

[4] X. Dong and T.H. Lai. "Distributed Dynamic Carrier Allocation in Mobile Cellular Networks: Search vs. Update". *Proceedings of 17 International Conference on Distributed Computing Systems*, 1997.

[5] S.M. Elnoubi, R. Singh, and S.C. Gupta. "A New Frequency Channel Assignment Algorithm in High Capacity Mobile Communication Systems". *IEEE Trans. Vehicular Tech.*, 31(3) 1982.

[6] D.J. Goodman. "Cellular Packet Communication". *IEEE Transactions on Communication*, pages 1272–1280, August 1990.

[7] T.J. Kahwa and N.D. Georganas. "A Hybrid Channel Assignment Scheme in Large-scale, Cellular-structured Mobile Communication Systems". *IEEE Transactions on Communication*, 26(4) 1978.

[8] I. Katzela and M. Naghshineh. "Channel Assignment Schemes or Cellular Mobile Telecommunication Systems: A Comprehensive Surbey". *IEEE Personal Communications*, pages 10–31, June 1996.

[9] L. Lamport. "Time, Clocks and Ordering of Events in Distributed Systems". *Comm. of the ACM*, July 1978.

[10] V.H. Macdonald. "Advanced Mobile Phone Service: The Cellular Concept". *Bell System Technical Journal*, pages 15–41, January 1979.

[11] S. Nanda and D.J. Goodman. "Dynamic Resource Acquisition: Distributed Carrier Allocation for TDMA Cellular Systems". *Proc. GLOBECOM'91*, pages 883–889, December 1991.

[12] R. Prakash, N. Shivaratri, and M. Singhal. "Distributed Dynamic Channel Allocation for Mobile Computing". *Proc. of 14th ACM Symp. on Principles of Distributed Computing*, pages 47–56, 1995.

[13] G. Ricart and A.K. Agrawal. "An Optimal Algorithm for mutual Exclusion in Computer Networks". *Communication of the ACM*, January 1981.

[14] J. Tajima and K. Imamura. "A Strategy for Flexible Channel Assignment in Mobile Communication Systems". *IEEE Trans. Vehicular Tech.*, 37 1988.

[15] M. Zhang and T.S. Yum. "Comparisons of Channel Assignment Strategies in Cellular Mobile Telephone Systems". *IEEE Trans. Vehicular Tech.*, pages 211–215, Nov. 1989.