Journal of Parallel and Distributed Computing **60**, 451–473 (2000)
doi:10.1006/jpdc.1999.1614, available online at http://www.idealibrary.com on IDEAL®

An Adaptive Distributed Channel Allocation Strategy for Mobile Cellular Networks

Guohong Cao

Department of Computer Science and Engineering, The Pennsylvania State University, University Park, Pennsylvania 16802
E-mail: gcao@cse.psu.edu

and

Mukesh Singhal

Department of Computer and Information Science, The Ohio State University, Columbus, Ohio 43210
E-mail: singhal@cis.ohio-state.edu

Received January 15, 1999; revised August 15, 1999; accepted December 15, 1999

A channel allocation algorithm includes a channel acquisition algorithm and a channel selection algorithm. Most of the previous work concentrates on the channel selection algorithm since early channel allocation algorithms simply use a centralized channel acquisition algorithm, which depends on a *mobile switching center (MSC)* to accomplish channel acquisition. Recently, distributed channel acquisition algorithms have received considerable attention due to their high reliability and scalability. There are two approaches to designing distributed channel acquisition algorithms: *search* and *update*. 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 propose a novel distributed channel acquisition algorithm, which is a significant improvement over both approaches. Also, we identify two guiding principles in designing channel selection algorithms and propose an algorithm which has low call blocking rate and low intra-handoff overhead. By integrating the channel selection algorithm into our channel acquisition algorithm, we get a complete distributed channel allocation algorithm. By keeping the borrowed channels, the channel allocation algorithm makes use of the temporal locality and adapts to the network traffic; i.e., free channels are transferred to hot cells to achieve load balance. Simulation results show that our channel allocation algorithm significantly outperforms

0743-7315/00 $35.00

the search approach and the update approach in terms of call blocking rate, message complexity, and acquisition delay.

Key Words: distributed channel allocation; channel borrowing; cellular networks.

1. INTRODUCTION

Cellular communication networks divide a geographical area into smaller hexagonal regions, called cells [9]. 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 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, 15]; otherwise, the communication sessions will interfere with each other, which is referred to as *channel interference*.

A channel is *available* for a cell if its use in the cell does not interfere with that of other cells. 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 message complexity and 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 an *MSS* to allocate a channel. The performance of the channel selection algorithm is measured by the *call blacking 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 host mobility.

1.1. Channel Selection Algorithms

There are three types of channel selection algorithms: fixed, flexible, and dynamic [12]. In the fixed strategies [14], a set of channels are permanently allocated to each cell, which is allowed to use the allocated channels and no others. In the dynamic strategies [3, 7], a cell may use any channel that will not cause channel interference. Channels are not preallocated to cells, but assigned on a dynamic basis. Typically, each channel is associated with a priority, and when a cell needs a channel, it picks the available channel with the highest priority. The channel is later returned to the system when it is no longer needed by the cell. Flexible strategies [19] combine the aspects of both fixed and dynamic strategies. Where each cell is

allocated a fixed set of permanent channels and a number of flexible channels are set aside to be dynamically allocated to cells upon requests.

Among these three strategies, dynamic strategies have been the focus of recent research [1, 7, 15]. Thus, we only consider *dynamic channel selection (DCS)* strategies. With DCS strategies, a cell may use any channel that will not cause channel interference. Typically, each channel is associated with a priority; when a cell needs a channel, it picks the available channel with the highest priority. Thus, various DCS strategies differ from one another in the way priorities are assigned to channels. There are three ways to assign channel priorities: static, dynamic, and hybrid. In a static-priority strategy such as the geometric strategy [1], each channel in each cell is assigned a fixed priority that does not change over time. In a dynamic-priority strategy such as the two-step strategy [6], the channel priority is dynamically computed. A hybrid-priority scheme [8, 20] is something in between: the channel priority is calculated as a static base-priority plus a dynamic adaptive-priority.

In the geometric strategy [1], each cell is assigned some channels as primary channels based on *a priori*. These primary channels are prioritized. During channel acquisition, a cell acquires the available primary channel with the highest priority. If none of the primary channels is available, the cell borrows a channel from its neighbors according to some fixed-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.

In the borrowing with directional channel-locking (BDCL) strategy [20], when a cell needs to borrow a channel, it borrows the channel with the lowest priority from the "richest" interference neighbor, 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 Nanda–Goodman strategy [15], when a cell borrows a channel, it selects a channel which will interfere with a smaller number of neighbors. When a cell releases a channel, it releases a channel which will make itself available in more interference neighbors.

The two-step strategy [6] combines the geometric strategy and the Nanda–Goodman strategy. In this approach, by using resource planing, the primary channels can be optimally utilized. At the same time, when a cell borrows a channel, it interferes with a minimum number of neighbors. However, since it does not consider the "richness," the lender may soon use up its channel and borrow channels again, and then the advantage of resource planning is missing.

All these algorithms depend on a *mobile switching center* (*MSC*) to accomplish channel acquisition, and then their associated channel acquisition algorithms are referred to a *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.

1.2. Channel Acquisition Algorithms

Recently, distributed channel acquisition algorithms [7, 17] have received considerable attention because of their high reliability and scalability. In this

approach, an *MSS* communicates with other *MSS*s directly to find the available channels and to ensure that assigning a channel does not cause interference with other cells. In general, there are two approaches to designing distributed channel acquisition algorithms: *search* [17] and *update* [7]. 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 to find out whetter 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 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 identify two guiding principles in designing channel selection algorithms. Following these principles, we propose a channel selection algorithm with which to further improve the performance of the two-step strategy by considering the "richness" and the interference property. Then, we propose a novel distributed acquisition algorithm whose message complexity is similar to that of the search approach and whose acquisition delay is similar to that of the update approach. By integrating the channel selection algorithm into our channel acquisition algorithm, we get a complete distributed channel allocation algorithm. By keeping the borrowed channels, the channel allocation algorithm makes use of the temporal locality and adapts to the network traffic; i.e., free channels are transferred to hot cells to achieve load balance. Detailed simulation experiments are carried out to evaluate our proposed methodology. Our algorithm outperforms centralized approaches such as the geometric strategy [1] and the two-step strategy [6] in terms of call blocking rate and intra handoff overhead under uniform and non-uniform traffic distributions. Our algorithm outperforms distributed algorithms such as the search approach [17] and the update approach [7] in terms of call blocking rate, message complexity, and acquisition delay.

The rest of this paper is organized as follows. Section 2 presents the system model. In Section 3, we propose a channel selection algorithm. Section 4 presents a distributed channel acquisition algorithm, combines it with our channel selection algorithm, and compares 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

Most channel selection strategies [1, 6, 7] require *a priori* knowledge of channel status in order to achieve better channel reuse. For instance, in the channel allocation strategies [8, 11, 15], each cell is allocated a set of "nominal" channels beforehand; in the geometric strategy [1], each cell must know its "first-choice" channels prior

to any channel acquisition. We call the process of assigning special status to channels *resource planning* [6, 7].

2.1. Resource Planning

The following is a resource planning strategy which has three rules:

1. Partition the set of all cells into a number of disjoint subsets $G_0, G_1, ..., G_{k-1}$, such that any two cells in the same subset are separated by at least a distance of $D_{\min}$. Accordingly, partition the set of all channels into k disjoint subsets: $P_0, P_1, ..., P_{k-1}$.

2. The channels in P_i $(i = 0, 1, ..., k-1)$ 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 channel is 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 1. Given a cell C_i, the set of *interference neighbors* of C_i, denoted by IN_i, is

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

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.

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

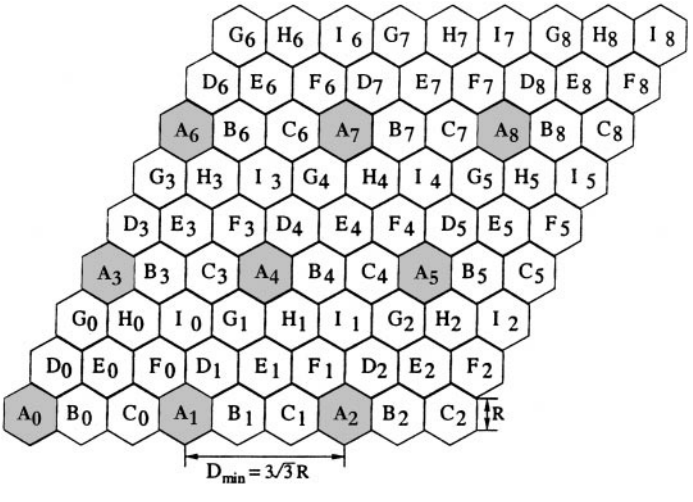

FIG. 1. An optimal partition.

orthogonal to our discussion, but we require that the partition satisfy the following property according to the resource planning model, which is obvious:

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

As shown in Fig. 1, R is the cell radius, and $D_{\min}$ is the minimum channel reuse distance. Cells are divided into nine subsets $G_A, G_B, ..., G_I$. Cells in $G_A = \{C_{A_i} \mid 0 \leqslant i \leqslant 8\}$ can use the same channel without interference. Because 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. When the distance between two cells are exactly $D_{\min}$, these cells are called *co-channel* cells. In our optimal partition model, cells in each subset are co-channel cells. For example, C_{A_1}, C_{A_2}, C_{A_3}, C_{A_5}, C_{A_6}, and C_{A_7} are co-channel cells of C_{A_4}.

2.2. Handoff and Intrahandoff

An *MH* 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 *interhandoff*) [14]. During a handoff, an *MH* releases its current channel to its previous *MSS* and is assigned a channel by the new *MSS*.

To achieve better channel reuse, *intrahandoff* (or a channel switch) may be necessary [2, 7]. In an intrahandoff operation, an *MH* releases its current channel and is assigned a new channel within the same cell. The motivation behind intrahandoff can be understood by an example. In Fig. 1, suppose cell C_{F_1} borrows a channel $r1$ from A_1 and assigns it to a mobile host MH_i. 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 intrahandoff from $r1$ to $r2$ by MH_i 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 intrahandoff is, of course, the overhead. Fortunately, most of the channel selection strategies do not demand many intrahandoffs [2, 7]. Thus, intrahandoff may be necessary for better channel reuse.

3. THE CHANNEL SELECTION ALGORITHM

Similar to the geometric strategy [1] and the update approach [7], our channel selection algorithm makes use of the 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 one of the primary channels is available, the cell borrows a channel from its neighbors according to some priority assignment approach. Before presenting our priority assignment strategy, we make two observations and identify two guiding principles in designing channel selection algorithms.

Observation 1. In Fig. 1, after C_{H_1} borrows a channel $r1$ from C_{A_4} ($r1$ is a primary channel of C_{A_4}), C_{A_1}, C_{A_2}, C_{A_4}, and C_{A_5} cannot use $r1$ due to channel interference. Thus, the borrowing of $r1$ interferes with four cells. Suppose C_{F_1} borrows $r2$ from C_{A_4}. Later, C_{H_4} runs out of channels. If C_{H_4} borrows a channel $r3$

from C_{A_4}, the borrow of $r3$ interferes with four cells: C_{A_4}, C_{A_5}, C_{A_7}, and C_{A_8}. However, if C_{H_4} borrows channel $r2$ from C_{A_4}, the borrowing of $r2$ only interferes with two *new* cells: C_{A_7} and C_{A_8}. (C_{A_4} and C_{A_5} cannot use $r2$ since C_{F_1} borrowed $r2$.) Later, if C_{F_4} wants to borrow a channel from C_{A_4}, it cannot borrow $r1$ and $r2$ due to channel interference, but the borrowing of any other channel may interfere with four cells. Note that if C_{H_4} borrows $r1$ instead of $r2$ from C_{A_4}, C_{F_4} can borrow $r2$ and only interferes with two new cells C_{A_7} and C_{A_8}. Based on this observation, we identify the following principle.

PRINCIPLE 1. *When a cell borrows a channel, it should select a channel which interferes with a smaller number of lenders. Also, if possible, the selected channel should be the same channel borrowed by its co-channel cells.*

Observation 2. In Fig. 1, suppose C_{H_1} borrows channel $r1$ from C_{A_4}. As a result, C_{A_1}, C_{A_2}, C_{A_4}, and C_{A_5} cannot use $r1$. Suppose C_{A_4} runs out of primary channels just after it lends channel $r1$ to C_{H_1}. Then, it needs to borrow a channel from other cells, which may interfere with four cells. Therefore, C_{H_1} should only borrow channels from the "richest" interference neighbor, 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. Based on this observation, we identify the following principle.

PRINCIPLE 2. *A cell should try to borrow a channel from the "richest" interference neighbor.*

We propose a priority assignment strategy to integrate these two principles. Let the cells be partitioned into k disjoint optimal reuse patterns $G_0, G_1, ..., G_{k-1}$, as defined in Section 2. Without loss of generality, we assume that there are a total of $k * N$ channels numbered $0, 1, ..., k * N - 1$ which are evenly divided into k subsets: $P_0, P_1, ..., P_{k-1}$ (this assumption is not essential and is made only for ease of presentation). A cell C_i is assigned N channels numbered $\min_i, \min_i + 1, ..., \min_i + N - 1$. Note that $\forall i \forall j \ (C_i \in G_k \wedge C_j \in G_k \Rightarrow \min_i = \min_j)$. To present the priority assignment strategy, we introduce the following notations:

- A_i: the set of currently known available channels at C_i.
- $PC(r)$: the set of primary cells of r.
- $CO_i(r)$: the number of co-channel cells of C_i that are borrowing channel r.
- $I_i(r)$: the set of cells to which C_i has lent channel r.
- δ: when a cell needs to borrow a channel, if possible, it should not borrow channels from those cells whose available channels are less than the threshold δ; δ is a system tuning factor (see Section 5.1).

DEFINITION 3. Given a cell $C_i \notin PC(r)$, the "richness" of channel r relative to C_i, denoted by $RH_i(r)$, is measured as the minimum number of primary channels available in the interference primary cells of r relative to C_i:

$$RH_i(r) = \min\{ |A_j| \mid C_j \in PC(r) \cap IN_i \}.$$

DEFINITION 4. Given a cell $C_i \notin PC(r)$, before the borrowing of channel r, the number of lenders which have lent r to some cells other than C_i, denoted by $BI_i(r)$, is defined as

$$BI_i(r) = |\{C_j \mid C_j \in PC(r) \cap IN_i \wedge I_j(r) \neq \varnothing\}|.$$

Based on these definitions, the priority of a channel r relative to cell C_i is defined as

$$\mathscr{P}_i = \begin{cases} m - (r - \min_i) & \text{if} \quad C_i \in PC(r) \\ (r - \min_k) + o * (CO_i(r) + BI_i(r)) * (RH_i(r) - \delta) & \\ & \text{if} \quad C_i \notin PC(r) \wedge C_k \in PC(r), \end{cases} \tag{1}$$

where $m \gg o \geqslant N$, e.g., $m = 11 * o * o$, $o = N + 1$.

From Eq. (1), the primary channels in a cell have the highest priority since m is a significantly large number. For secondary channels, the priority is determined by Principles 1 and 2; if two channels have the same interference property and "richness," the channel with the higher number has higher priority. Initially, we do not consider the relative importance of Principles 1 and 2; this strategy could be extended by changing the relative importance of these two principles. We found (by simulation) that a lender should not lend any channel to others when its available channels are lower than δ if it is possible, i.e., the borrower can borrow channels from other cells.

3.1. Reducing the Overhead of Intrahandoff

In Fig. 1, suppose cell C_{A_1} has two primary channels $r1$ and $r2$. C_{A_1} is using $r2$, while cells C_{A_2}, C_{A_4}, and C_{A_5} are using $r1$. Even though $r1$ is available in C_{A_1} and $r2$ is available in cells C_{A_2}, C_{A_4}, and C_{A_5}, neither $r1$ nor $r2$ can be borrowed by C_{H_1}. If an intrahandoff is performed (i.e., C_{A_1} releases $r2$ and uses $r1$), C_{H_1} can borrow $r2$. Thus, when a cell has several available primary channels, it acquires the highest priority channel and releases the lowest priority channel. If a newly available primary channel has higher priority than some used primary channels, an intrahandoff is performed.

Since intrahandoffs increase system overhead, we use the following approach to reduce the number of intrahandoffs. If an intrahandoff is between two channels whose channel sequence numbers are smaller than a threshold θ, this intrahandoff can be avoided. The reason is as follows. According to our channel priority assignment strategy, a cell uses small sequence number channels and lends high sequence number channels to other cells. For a cell C_i, if both intrahandoff channels have small sequence numbers, C_i is more likely to have a large number of available channels, and it has a low probability for other cells to select the intrahandoff channels to borrow.

In our algorithm, for a cell C_i, the threshold θ is set to be $\min_i + N/2$. Certainly, a fine grain tuning may further reduce the number of intrahandoffs, but it may also increase the call blocking rate.

4. AN ADAPTIVE DISTRIBUTED CHANNEL ALLOCATION ALGORITHM

In this section, we investigate the fundamental difference between the search and the update approaches. Then, we propose a distributed channel acquisition algorithm and combine it with our channel selection algorithm to get a complete channel allocation algorithm. Finally, we compare the complete channel allocation algorithm with the search and the update approaches in terms of message complexity and acquisition delay.

4.1. Search vs Update

Both the search and the update approaches time stamp control messages using Lamport's logical clocks [13] to determine the priority of requests.

4.1.1. The Search Approach

In the search approach [17], when a cell (the borrower) needs to borrow a channel, it changes to *search mode* and sends *request* messages to each cell in IN_i. When a cell (the lender) receives a *request* from the borrower, if the lender is not in the search mode or it is in the search mode but its request has higher timestamp (lower priority) than the borrower, the lender sends a *reply* message to the borrower which contains information about its used channels; otherwise, the lender defers the *reply* (similar to [18]). After the borrower has received all the *reply* messages from each cell in IN_i, it computes the available channels and picks one, say r, from them. The borrower sends *confirm* messages to the lenders of r. If all lenders reply *agree*, the borrower can use r; otherwise, the borrower picks another available channel and repeats the process. If there is no available channel left, the call request is failed. When a channel r is borrowed, the lender marks r as an *interference channel*, and it cannot use r until r is returned by all borrowers.

4.1.2. The Update Approach

In the update approach [7], a cell maintains information about the available channels. When a cell (the borrower) needs to borrow 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 time stamp; otherwise, it replies with an *agree*. If the borrower has received *agree* messages from all the cells in IN_i, it notifies them that it has successfully acquired channel r; otherwise, it picks another available channel and repeats the process. When the borrower finishes the use of the borrowed channel, it sends a *release* to each cell in IN_i.

4.1.3. 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. In the search approach, a cell needs to confirm a selected channel, which doubles the acquisition delay compared to the update approach.

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, an intrahandoff 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 cell releases a channel. Therefore, many good channel selection strategies cannot be supported in the search approach. Moreover, the search approach [17] 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 which does not lock the borrowed channel.

4.2. A Distributed Channel Acquisition Algorithm

4.2.1. Reducing the Acquisition Delay

In the search approach, a cell has to *confirm* a selected channel with the lenders since a lender may assign that channel to a new call immediately after it sends 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 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, as in 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.

4.2.2. 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 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 $I_i(r) \neq \emptyset$, r is an interference channel of C_i, in which case C_i cannot use r but can lend it to other cells.

4.2.3. The Channel Acquisition Algorithm

In the distributed channel acquisition algorithm, if a cell C_i has an available primary channel r, it can use r immediately unless an interference neighbor is in the search mode ($send_i \neq \emptyset$). When $send_i \neq \emptyset$, C_i can only use the channels in $reserved_i$ or wait for $send_i = \emptyset$. 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 Fig. 2. If no channel is selected from the reserved channels, five types of messages are exchanged among *MSS*s 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. In this case, the cell can just pick more channels and assign them to these call requests. For simplicity, this is not explicitly presented in the algorithm.

4.3. 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 and depends on the condition of $reserved_i$, there are three possibilities.

Case 1.1: $send_i = \emptyset$. 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 $reply(U_i, reserved_i)$, $r \in U_i \Rightarrow r \notin A_j$ according to Step C.1. Then C_j cannot acquire r.

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 $A_i = P_i - U_i - \{r \mid I_i(r) \neq \emptyset \wedge r \in P_i\}$ 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) $send_i = \emptyset$: 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) $send_i \neq \emptyset$: 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 clears its search mode; otherwise, it does the follows:

(B.1) $send_i = \emptyset$: 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) $send_i \neq \emptyset$: 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:

$$A_i = S - U_i - \{r \mid I_i(r) \neq \emptyset \wedge r \in P_i\} - \bigcup_{k \in IN_i} U_k$$

(C.2) If A_i is empty, C_i drops the call, clears 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 clears 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 $transfer(r)$ from C_j, 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.

FIG. 2. The distributed channel acquisition algorithm.

Case 1.2: $send_i \neq \emptyset \wedge r \in reserved_i$. 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: $send_i \neq \emptyset \wedge r \notin reserved_i$. 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 1.3.2: $C_j \notin send_i$. If C_j's request arrives before $send_i = \emptyset$, 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.

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 a smaller time stamp 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 $reply(U_i, reserved_i)$, $r \in U_i \Rightarrow r \notin A_j$ 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 are totally ordered by their time stamps. An *MSS* in search mode sends *reply* messages to all *requests* with a lower time stamp and defers others. As the same ordering of channel requests is seen by all the *MSS*s, there is no circular deferring of *reply*s among the *MSS*s.

Because 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 with either 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. ▮

4.4. The Complete Channel Allocation Algorithm

Most of the existing DCS strategies [4, 5, 8, 10, 11, 12, 16, 19] need up-to-date information to calculate channel priority. This can be easily implemented in centralized algorithms, since an *MSC* monitors every release and acquisition of channels, and thus has up-to-date information. However, in a distributed channel allocation algorithm, due to unpredictable message delay, obtaining the instantaneous global state information is practically impossible. Thus, we can only get the approximately up-to-date information by increasing the message overhead. To combine the channel selection algorithm with our distributed channel acquisition algorithm and without significantly increasing the message overhead, we make the following modifications to our algorithm:

- Whenever a cell borrows a channel, it asks its six surrounding co-channel cells for the channels they have borrowed. This information is used to calculate channel priority using Eq. (1).

- To make use of locality, a cell does not return the *borrowed* channel immediately after its use. Instead, it keeps the borrowed channel. Thus, there are two kinds of borrowed channels: *used-borrowed channels* and *available-borrowed channels*. Used-borrowed channels are counted as used channels. Available-borrowed channels are counted as available channels and can be lent to other cells. For example, in Fig. 1, suppose C_{H_1} borrows a channel r from C_{A_4}. After using r, it keeps r as an available-borrowed channel. Later, C_{B_4} wants to borrow r from C_{A_4}. C_{B_4} knows that r is an available-borrowed channel based on the collected channel information

from its interference neighbors. Thus, C_{B_4} only needs to confirm with C_{H_1} and C_{A_7} before using r. It does not need to receive permission from C_{A_2}, C_{A_4}, and C_{A_5} since these cells have granted the permission to C_{H_1}. If C_{H_1} agrees, it sends *agree* to C_{B_4} and sends *release* to C_{A_1}. If C_{B_4} also gets *agree* from C_{A_7}, it sends a special message to notify C_{A_2}, C_{A_4}, and C_{A_5} that C_{H_1} has released them and C_{B_4} is locking them. Receiving this special message, C_{A_2} deletes C_{H_1} from $I_{A_2}(r)$ and adds C_{B_4} to $I_{A_2}(r)$. C_{A_4} and C_{A_5} act similarly.

When a cell knows that its lender's available channels are less than a threshold η (determined in Section 5.1), it releases the borrowed channels (from that lender) if they are not being used. Otherwise, it should perform an intrahandoff to release that channel if possible (it is not possible when there it no available channel). Whenever a communication session (or a call) is over, the cell checks whether it has too many available channels (e.g., 10% of the number of assigned primary channels); if so, it releases some available-borrowed channels.

- There are two approaches to reducing the message overhead. In Approach 1, we modify Steps 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. Then, a cell keeps the up-to-date information for calculating the channel priority of its interference neighbors. This approach reduces the message overhead compared to the update approach, since the number of borrowers is very small compared to the number of interference neighbors. In Approach 2, a cell only notifies the cells that have borrowed channels from it when its available channels are less than η' $(\eta' > \eta)$.

The disadvantage of Approach 2 is that the borrower may not know the up-to-date information. The advantages are low message overhead and low intrahandoff overhead. Knowing the up-to-date information is only helpful when releasing the borrowed channels. Because we want to make use of locality by keeping borrowed channels, and because a borrowed channel is released when its lender's available channels are lower than η, it may not be necessary to know the up-to-date information of the lender considering the high message overhead. Thus, we implemented Approach 2 in our algorithm with $\eta' = 10\%$ of the number of assigned primary channels.

By these modifications, our distributed channel allocation algorithm significantly reduces message complexity. Moreover, our algorithm adapts to the network traffic; i.e., free channels are transferred to hot cells to achieve load balance.

4.5. Performance Analysis and Comparison

We analyze the performance of our channel allocation algorithm and compare it to the search [17] and the update approaches [7]. 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 $3n + 12 + n_p$, which includes $n + 6$ (there are six co-channel cells) *request* and *reply*

TABLE 1

Comparison of the Update, the Search, and Our Approach

Approach	Message complexity	Acquisition delay
Search	$\alpha' * (2 * n + 3 * n_p * (1 + m'))$	$2 * T * (2 + m') + T'_d$
Update	$2 * n + \alpha'' * (3 * n_p * m'' + 2 * n_p)$	$2 * T * (1 + m'') + T''_d$
Our approach	$\leqslant \alpha''' * (3 * n + 12 + 3 * n_p * m''' + n_p) + n_u$	$2 * T * (1 + m''') + T'''_d$

messages, $n - n_p$ *finish*, and n_p *transfer* and *release* messages. If a cell confirms m ($m \geqslant 1$) times before it acquires a channel, the number of messages per secondary channel acquisition is $3n + 12 + n_p + 2 * m * n_p + (m-1) * n_p$, 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*.

Let n_u denote the number of *update* messages needed in our algorithm, $\alpha(\alpha', \alpha'', \alpha''')$ denote the percentage of secondary channel acquisition, $m(m', m'', m''')$ 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 messages 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. Also, $\alpha''' < \alpha'' < \alpha'$. From Table 1, our algorithm almost cuts the secondary channel acquisition delay to half compared to the search approach. When the channel request load is low, it is usually 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, and 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 n_u is 0 if we use the geometric strategy or the update approach as the underlying channel selection algorithm). Thus, our approach significantly reduces the message complexity compared to the update approach, whose message complexity is always larger than $2 * n$.

There are some similarities between the search approach and the proposed algorithm; e.g., both keep the borrowed channel and both halve low message complexity. However, there are significant differences between these two approaches. First, by reserving some channels during channel acquisition, the proposed algorithm cuts the delay by almost half. Second, in the search approach, a cell never returns the borrowed channel. After a cell borrows a channel, it becomes the owner of the borrowed channel. Due to this ownership change, it is impossible to make use of resource planning in the search approach. As a result, a cell randomly chooses a channel without considering optimal channel reuse, which results in a high call blocking rate. Because cells randomly borrow channels from their neighbors, after a borrower borrows a channel from a lender, the lender may run out of channel

and borrow channels from its neighbors (even the borrower), which results in a problem similar to context switching. In our algorithm, this problem is avoided by following Principle 2, where a cell only borrows channels from the richest neighbors. By following Principles 1 and 2, our algorithm has lower secondary channel acquisition rate than the search approach. As a result, our algorithm has lower message complexity and lower acquisition delay. Third, The search approach does not have intrahandoff—as explained in Section 2, intrahandoff can significantly reduce call blocking rate—and thus, our algorithm has a lower call blocking rate than the search approach. Note that the search approach also has some advantages, such as a simple and no intrahandoff overhead.

5. SIMULATION RESULTS

We studied the performance of the proposed channel allocation algorithm, the search approach [17], the update approach [7], the geometric strategy [1], and the two-step strategy [6] using extensive simulations. 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.

5.1. Simulation Parameters

The simulated cellular network is a wrapped-around layout with $12 * 12$ cells. The total number of channels in the system is 396. If a fourth-power law attenuation is assumed [1, 14], the signal-to-interference ratio is given by $[S/I]_{\min} = [(D_{\min}/R) - 1]^4/6$. With $D_{\min} = 3\sqrt{3}\,R$, $[S/I]_{\min} \approx 17$ dB, which is a reasonable value in practice. Thus, we choose $D_{\min} = 3\sqrt{3}\,R$, and then each cell is assigned $396/9 = 44$ channels. Since a single bit is enough to represent whether or not the channel is used, the control message in our algorithm is very small compared to the header. Considering that an *MSS* may not respond immediately to incoming messages, we assume that the average one-way communication delay between two *MSS*s is 2 ms, 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 arrival time, mean service time, and mean interhandoff time, all assumed to be negative exponentially distributed.

TABLE 2

Simulation Parameters for Uniform Traffic Distribution

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

TABLE 3

Simulation Parameters for Nonuniform Traffic Distribution

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

Under nonuniform 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 interhandoff rate. On the contrary, a cell in the hot state is characterized by high arrival rate and low interhandoff 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 the time when it sends out a *reply* to the time when it receives 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 our simulation. Similarly, a lender only needs $2 * T$ to ask the borrowers to return the borrowed channels, and the lender has a very low probability (0.00003) of receiving another call during the $2 * T$. Hence, we choose $\eta = 1$ in our simulation.

We use simulation to determine the value of parameter δ. We consider δ to be 0, 5% ($\delta = 2$), 10% ($\delta = 5$), 20% ($\delta = 9$) of the primary channels. From Fig. 3, we can see that $\delta = 2$ has the best performance and $\delta = 0$ has the worst performance. However, the difference is not too much. Moreover, sometimes $\delta = 2$ performs worse than others; e.g., $\delta = 5$ performs better when the arrival rate is 650 under nonuniform distribution. When $\delta = 0$, even though the borrower only has one channel left, it may still lend the channel to other cells, in which case the lender is likely to run out of channel and borrow channels again. This can be solved by increasing δ

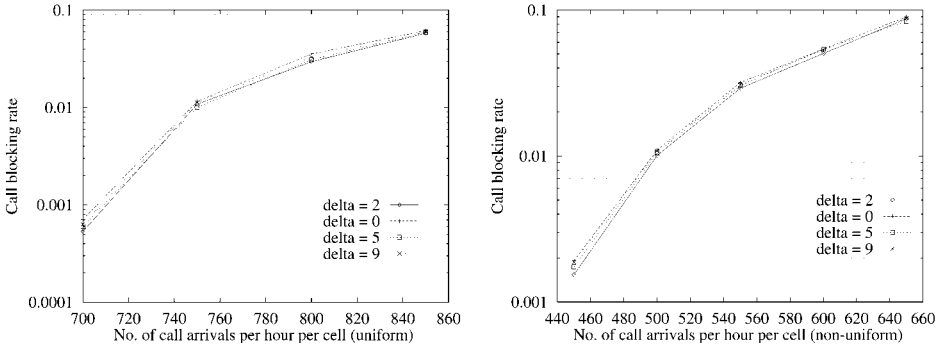

FIG. 3. Comparisons of call blocking rate.

to 2. However, further increasing δ performs worse since Principle 1 may not be adequately considered. In the following, we assume $\delta = 2$.

5.2. Simulation Results

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 when we analyze the message overhead and acquisition delay for the sake of simplicity.

5.2.1. Message Complexity per Channel Acquisition

As shown in Fig. 4, 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 4.5, the message, complexity in the search approach and the proposed algorithm is decided by the percentage of secondary channel acquisition. When the channel request is low, 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 4.5, our algorithm has higher message complexity $(3 * n)$ than the search approach $(2 * n)$, but Fig. 4 shows that our algorithm has

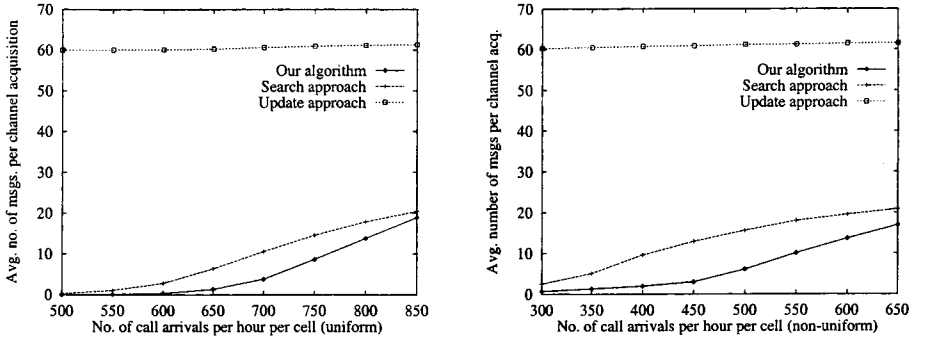

FIG. 4. Comparisons of message complexity.

lower message complexity than the search approach. This can be explained by the fact that both algorithms have different secondary channel acquisition percentage (see Fig. 5). Note that the acquisition of a local available-borrowed channel is not considered a secondary channel acquisition, since in this case, the borrower does not need to contact with its interference neighbors. From Fig. 5, we can see that the search approach has a higher secondary channel acquisition rate than does our algorithm, since the search approach does not consider channel reuse, i.e., a cell just randomly borrows a channel from its neighbors whenever it needs a channel. In our algorithm, frequency is optimally reused by resource planning. Also, by following Principles 1 and 2, cells in our algorithm borrow channels less frequently than cells in the search approach (see Fig. 5). Also, keeping the borrowed channels reduces the number of channel borrowing.

Under nonuniform traffic distribution, only some cells are in the hot state, and most of the borrowers are hot cells (cells in the hot state). In our approach, when a cell finishes using the borrowed channel, it keeps the channel. Then, free channels are transfered to these hot cells, and hence, new communication sessions in the hot cells can be supported without borrowing channels again. Under uniform traffic distribution, when the traffic load is high, most cells run out of channels; when the traffic load is low, most of them have free channels. Thus, the advantage of keeping channels under uniform traffic distribution is not very significant compared to that under nonuniform traffic distribution. This explains why our approach has a much lower secondary channel acquisition percentage than other approaches under non-uniform traffic distribution compared to uniform traffic distribution.

Under uniform traffic distribution, when the traffic load becomes very high, e.g., there are 850 call arrivals per hour per cell, it is more likely that the lenders have less than η available channels, and hence the borrowers cannot keep the borrowed channel. As a result, the secondary channel acquisition percentage in our approach increases much faster compared to other approaches at this point. Certainly, it is still lower than the secondary channel acquisition percentage in other approaches.

5.2.2. Acquisition Delay per Channel Transfer

As shown in Fig. 6, the search approach has the highest secondary channel acquisition delay since it needs to confirm every borrowed channel. Both our

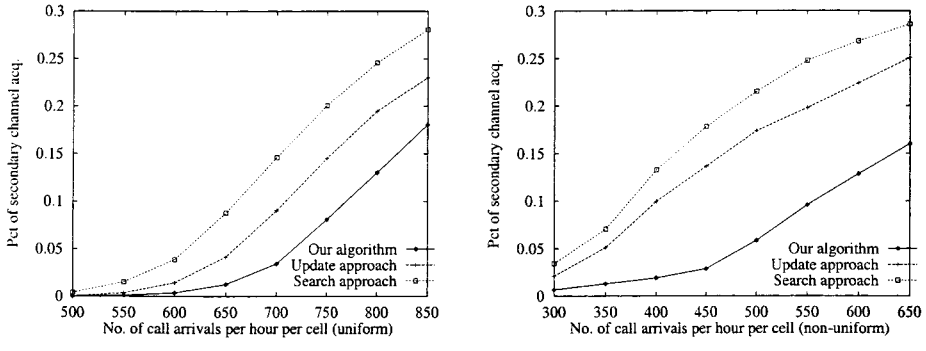

FIG. 5. Percentage of secondary channel acquisition.

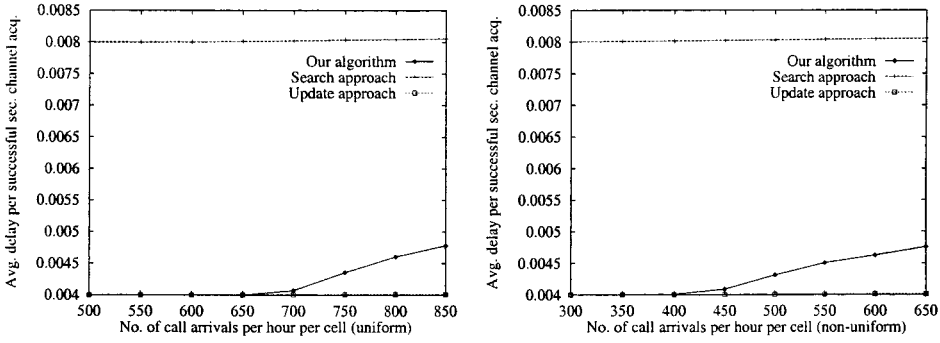

FIG. 6. Comparisons of secondary channel acquisition delay.

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 Fig. 6, we see that the chance of selecting a reserved channel is less than 20%. 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.

Because the primary channel acquisition delay is 0, and cells in our approach borrow channels less frequently than the search approach and the update approach (see Fig. 5), our algorithm has the lowest average acquisition delay among these approaches (see Fig. 7).

5.2.3. A Comparison of Call Blocking Rate

The blocking rate of our algorithm is compared with the geometric strategy, the search approach, the update approach, and the two-step strategy (Fig. 8). Because the geometric strategy, the update approach, the two-step strategy, and our algorithm are all based on the optimal resource planning model, the call blocking rates for these four approaches are not very different from one another, with our algorithm slightly outperforming the other three since only our algorithm follows both Principle 1 and Principle 2. In the geometric strategy, neither Principle 1 nor Principle 2 is followed.

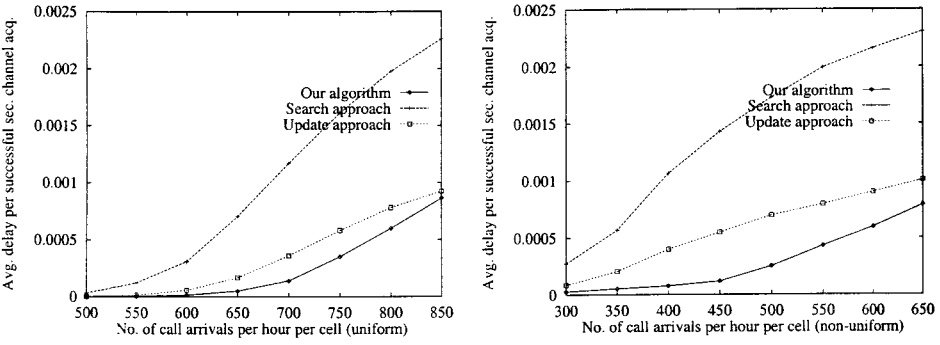

FIG. 7. Comparisons of average channel acquisition delay.

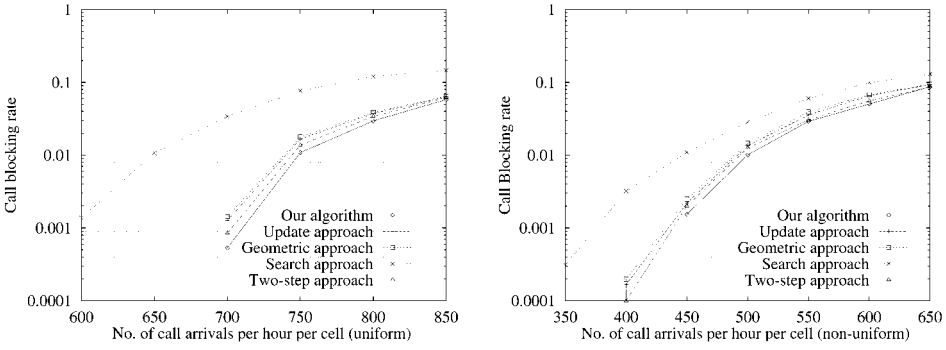

FIG. 8. Comparisons of call blocking rate.

The update approach only partially follows Principle 2, which makes it slightly outperform the geometric strategy. In the two-step strategy, only Principle 1 is considered, which makes it outperform the update approach. 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 that does not consider any channel reuse. Moreover, the search approach locks the borrowed channel during channel borrowing, which also reduces channel reuse.

Note that under high traffic load in the nonuniform distribution, the two-step strategy slightly outperforms our algorithm (650 calls per hour per cell). However, the two-step strategy is a centralized algorithm, and it has poor reliability and scalability. Also, most of time, it has a higher call blocking rate than ours. From Fig. 9, we can see that our algorithm has fewer intrahandoffs than the two-step strategy. In the two-step strategy, when a primary channel is released, there will be an intrahandoff if a call is using a borrowed channel. In our approach, because a borrowed channel can be temporarily saved locally, this kind of intrahandoff is not necessary. According to our simulation parameters, a call on average experiences three interhandoffs and each interhandoff has a channel release which is more likely associated with an intrahandoff. Thus, there are about three intrahandoffs for each call. Both the geometric strategy and the update approach have intrahandoff overhead similar to that of the two-step strategy. The search approach does not have intrahandoff overhead, which is one of the key reasons that it has a high call blocking rate.

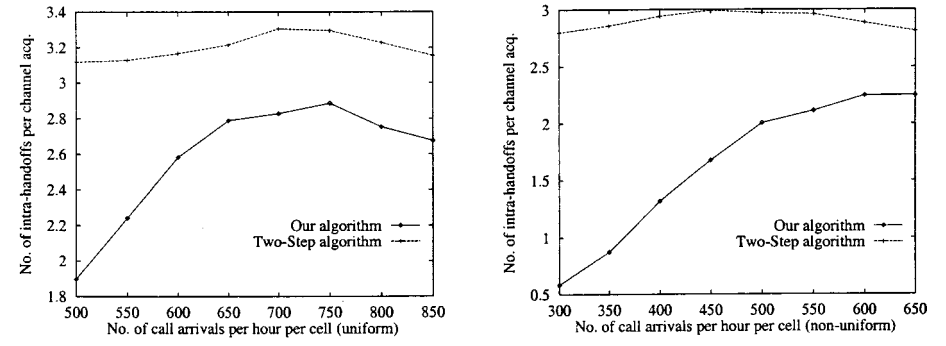

FIG. 9. Comparisons of intrahandoffs.

In the proposed algorithm, we use a threshold θ to reduce the number of intrahandoffs. As explained in Section 3, the value of θ can be fine tuned. Other methods can also be used to reduce the number of intrahandoffs. For example, a cell does not need to perform any intrahandoffs before it receives a channel borrowing *request*. When a cell receives a *request*, it requires necessary *MH*s (if there is any) to take an intrahandoff. Combined with the use of threshold θ, the number of intrahandoffs can be further reduced. However, the new method may increase the acquisition delay since the lender cannot reply the borrower until the necessary intrahandoff has been completed.

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 has high message complexity, or on the search approach, which has long acquisition delay and high call blocking rate. In this paper, we identified two guiding principles in designing channel selection algorithms and proposed a novel 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, cuts the secondary channel acquisition delay by almost half, and reduces the average channel acquisition delay by at least a factor of four. Compared to the update approach, our algorithm reduces the message overhead from 60 to 0 under low channel request load; under high channel request load, our algorithm reduces the message overhead by at least a factor of three. When compared to centralized channel allocation strategies, our algorithm outperforms them in terms of call blocking rate and intrahandoff overhead.

REFERENCES

1. A. Baiocchi, F. D. Priscoli, F. Grilli, and F. Sestini, The geometric dynamic channel allocation as a practical strategy in mobile networks with bursty user mobility, *IEEE Trans. Veh. Tech.* **44**, 1 (Feb. 1995), 14–23.
2. R. Beck and H. Panzer, Strategies for handover and dynamic channel allocation in micro-cellular mobile radio systems, *IEEE Veh. Tech. Conf.* **38**, 2 (May 1989), 668–672.
3. D. Cox and D. Reudink, Increasing channel occupancy in large scale mobile radio systems: Dynamic channel reassignment, *IEEE Trans. Veh. Tech.* **22**, 4 (Nov. 1973), 218–222.
4. S. K. Das, S. K. Sen, and R. Jayaram, A dynamic loadbalancing strategy for channel assignment using selective borrowing in cellular mobile environments, *ACM/Baltzer Wireless Networks* (*WINET*) **3**, 5 (1997), 333–347.
5. S. K. Das, S. K. Sen, and R. Jayaram, A novel load balancing scheme for the tele-traffic hot spot problem in cellular network, *ACM/Baltzer Wireless Networks* (*WINET*) **4**, 4 (1998), 325–340.
6. X. Dong and T. H. Lai, An efficient priority-based dynamic channel allocation for mobile cellular networks, *in* "IEEE INFOCOM," 1997.
7. X. Dong and T. H. Lai, Distributed dynamic carrier allocation in mobile cellular networks: Search vs. update, *in* "Proceedings, International Conference on Distributed Computing Systems," pp. 108–115, May 1997.

8. S. M. Elnoubi, R. Singh, and S. C. Gupta, A new frequency channel assignment algorithm in high capacity mobile communication systems, *IEEE Trans. Veh. Tech.* **31**, 3 (Aug. 1982), 125–131.
9. D. J. Goodman, Cellular packet communication, *IEEE Trans. Comm.* **38**, 8 (Aug. 1990), 1272–1280.
10. H. Jiang and S. Rappaport, Prioritized channel borrowing without locking: A channel sharing strategy for cellular communications, *IEEE/ACM Trans. Networking* **4**, 2 (Apr. 1996), 163–172.
11. T. J. Kahwa and N. D. Georganas, A hybrid channel assignment scheme in large-scale, cellular-structured mobile communication systems, *IEEE Trans. Comm.* **26**, 4 (Apr. 1978), 432–438.
12. I. Katzela and M. Naghshineh, Channel assignment schemes for cellular mobile telecommunication systems: A comprehensive survey, *IEEE Personal Comm.* **3**, 3 (Jun. 1996), 10–31.
13. L. Lamport, Time, clocks and ordering of events in distributed systems, *Comm. Assoc. Comput. Mach.* **21**, 7 (July 1978), 558–565.
14. V. H. Macdonald, Advanced mobile phone service: The cellular concept, *Bell System Technical J.* (Jan. 1979), 15–41.
15. S. Nanda and D. J. Goodman, Dynamic resource acquisition: Distributed carrier allocation for TDMA cellular systems, *in* "Proceedings, GLOBECOM'91," pp. 883–889, Dec. 1991.
16. L. Ortigozo-Guerrero and D. Lara-Rodriguez, A compact pattern with maximized channel borrowing strategy for mobile cellular networks, *in* "Proceedings, IEEE International Symposium on Personal, Indoor and Mobile Radio Communications," pp. 329–333, 1996.
17. R. Prakash, N. Shivaratri, and M. Singhal, Distributed dynamic channel allocation for mobile computing, *in* "Proceedings, 14th ACM Symposium on Principles of Distributed Computing," pp. 47–56, 1995.
18. G. Ricart and A. K. Agrawal, An optimal algorithm for mutual exclusion in computer networks, *Comm. Assoc. Comput. Mach.* **24** (1) (Jan. 1981), 9–17.
19. J. Tajima and K. Imamura, A strategy for flexible channel assignment in mobile communication systems, *IEEE Trans. Veh. Tech.* **37**, 2 (May 1988).
20. M. Zhang and T. S. Yum, Comparisons of channel assignment strategies in cellular mobile telephone systems, *IEEE Trans. Veh. Tech.* **38**, 4 (Nov. 1989), 211–215.

GUOHONG CAO received the B.S. degree from Xian Jiaotong University, Xian, China. He received the M.S. and Ph.D. degrees in computer science from the Ohio State University in 1997 and 1999, respectively. He was a recipient of a Presidential Fellowship at The Ohio State University. Since Fall 1999, he has been an assistant professor of computer science and engineering at Pennsylvania State University. His research interests include distributed fault-tolerant computing, mobile computing, and wireless networks.

MUKESH SINGHAL is a full professor of computer and information science at The Ohio State University, Columbus. He received a bachelor of engineering degree in electronics and communication engineering with high distinction from University of Roorkee, Roorkee, India, in 1980 and a Ph.D. degree in computer science from the University of Maryland, College Park, in May 1986. His current research interests include operating systems, database systems, distributed systems, mobile computing, high-speed networks, computer security, and performance modeling. He has published over 100 refereed articles in these areas. He has coauthored two books, titled "Advanced Concepts in Operating Systems" (McGraw–Hill, 1994) and "Readings in Distributed Computing Systems" (IEEE Computer Society Press, 1993). He is currently the program director of the operating systems and compilers program at the National Science Foundation.