Distributed Fault-Tolerant Channel Allocation for Cellular Networks

Guohong Cao, *Associate Member, IEEE,* and Mukesh Singhal, *Senior Member, IEEE*

***Abstract*—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 acquisition algorithms are centralized and rely 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. However, in these algorithms, a borrower needs to consult with its interference neighbors in order to borrow a channel. Thus, the borrower fails to borrow channels when it cannot communicate with any interference neighbor. In real-life networks, under heavy traffic load, a cell has a large probability to experience an intermittent network congestion or even a communication link failure. In existing distributed algorithms, since a cell has to consult with a large number of interference neighbors to borrow a channel, the failure rate will be much higher under heavy traffic load. Therefore, previous distributed channel allocation algorithms are not suitable for real-life networks. In this paper, we first propose a fault-tolerant channel acquisition algorithm which tolerates communication link failures and node (MH or MSS) failures. Then, we present a channel selection algorithm and integrate it into the distributed acquisition algorithm. Detailed simulation experiments are carried out in order to evaluate our proposed methodology. Simulation results show that our algorithm significantly reduces the failure rate under network congestion, communication link failures, and node failures compared to nonfault-tolerant channel allocation algorithms. Moreover, our algorithm has low message overhead compared to known distributed channel allocation algorithms, and outperforms them in terms of failure rate under uniform as well as nonuniform traffic distribution.**

***Index Terms*—Cellular networks, channel borrowing, distributed channel allocation, fault-tolerance, handoff.**

I. INTRODUCTION

CELLULAR communication networks divide a geographical area into smaller regions, called cells [11]. Each cell has a *mobile support station* (MSS) and a number of *mobile hosts* (MH's). 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 frequency spectrum available for civilian use is limited, the frequency channels have to 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], [17], because the communication sessions would interfere with each other. The *interference neighbors* of a cell C_i are cells whose geographic distance from C_i is less than $D_{\min}$.

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 an available channel using a channel allocation algorithm. A channel allocation algorithm includes 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 responsible for choosing a channel from a large number of available channels in order to achieve better channel reuse. The performance of a channel allocation algorithm is measured by the *failure rate* [1]. A call is said to have failed if there is no available channel when the call is being set up or when it is being handed over to another cell.

Channel selection algorithms [4], [9], [11]–[14], [18], [22] have been an active research topic for the last 30 years. However, most of these algorithms, which are referred to as *centralized* channel acquisition algorithms, rely on a *mobile switching center* (MSC) to accomplish channel acquisition. 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 requesting cells accordingly. Obviously, the centralized approach has low reliability and scalability.

Recently, distributed channel acquisition algorithms [8], [19] 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 guarantee that the channel assignment does not interfere with other cells. In general, there are two approaches in distributed channel acquisition algorithms: *Search* [19] and *Update* [8]. In the search approach [19], when a cell needs a channel, it searches all interference neighbors to find the set of currently available channels and then picks one according to the underlying channel selection strategy. In the update approach [8], 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 with its interference neighbors whether it can acquire the selected channel. Also, a cell informs its interference neighbors each time it acquires or releases a channel so that each cell always knows the available channels of its interference neighbors.

Manuscript received December 7, 1999; revised February 13, 2000.

G. Cao is with the Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA 16802 USA (e-mail: gcao@cse.psu.edu).

M. Singhal is with the Department of Computer and Information Science, The Ohio State University, Columbus, OH 43210 USA (e-mail: singhal@cis.ohio-state.edu).

Publisher Item Identifier S 0733-8716(00)05412-3.

Both approaches require the borrower to wait for the acknowledgments from its interference neighbors. Thus, a borrower cannot borrow a channel when it cannot communicate with anyone of them. Moreover, since many communication sessions such as handoff, audio, and video have time constraints, a long communication delay due to network congestion has the same effect as a communication link failures or node (MH or MSS) failures, where the borrower may fail to borrow a channel even though its neighbors still have many available channels. In real-life networks, under heavy traffic load, a cell has a large probability to experience an intermittent network congestion, a communication link failure, or a node failure. In these approaches [8], [19], since a cell has to consult with a large number of interference neighbors to borrow a channel, the failure rate will be much higher under heavy traffic load. Therefore, these approaches may not be suitable for real-life networks.

In this paper, we propose a fault-tolerant channel acquisition algorithm which tolerates communication link failures and node (MH or MSS) failures. In the proposed algorithm, the borrower does not need to receive a response from every interference neighbor. It only needs to receive responses from a small portion of them. Thus, as long as the borrower can communicate with a small portion of its interference neighbors, it can borrow channels from them. We also present a channel selection algorithm and integrate it into the distributed channel acquisition algorithm. Detailed simulation experiments are carried out to evaluate our proposed methodology. Simulation results show that our algorithm significantly reduces the failure rate under network congestion, communication link failures, and node failures compared to nonfault-tolerant channel allocation algorithms. Besides providing fault-tolerance, our algorithm reduces the message overhead compared to known distributed channel allocation algorithms, and outperforms them in terms of failure rate under uniform as well as nonuniform traffic distribution.

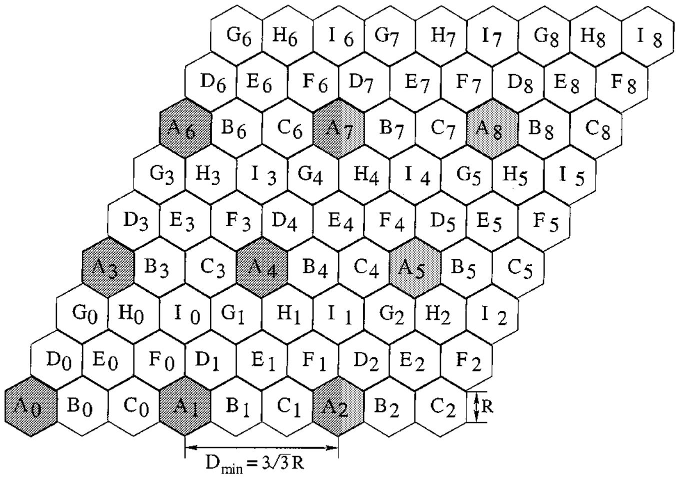
The rest of the paper is organized as follows. Section II presents the system model. In Section III, we propose a fault-tolerant channel acquisition algorithm, give correctness proofs, and propose some recovery techniques. Section IV presents a channel selection algorithm and integrates it into the distributed channel acquisition algorithm. Simulation results are presented in Section V. Section VI concludes the paper.

II. System Model

In cellular networks, a geographical area is divided into hexagonal cells [11]. Each cell is supported by an MSS in the center (we use the terms cell and MSS interchangeably). The MSS's are connected to each other by a static wired network.

A node (MH or MSS) may either crash or fail to send or receive messages. Communication links may fail by crashing or by failing to deliver messages. Combinations of such failures may lead to partition failures [6], where nodes in a partition may communicate with each other, but no communication can occur between nodes in different partitions.

To establish a communication session (or a call), an MH has to send a request to the MSS in its cell. The session is admitted



G6 H6 I6 G7 H7 I7 G8 H8 I8
D6 E6 F6 D7 E7 F7 D8 E8 F8
A6 B6 C6 A7 B7 C7 A8 B8 C8
G3 H3 I3 G4 H4 I4 G5 H5 I5
D3 E3 F3 D4 E4 F4 D5 E5 F5
A3 B3 C3 A4 B4 C4 A5 B5 C5
G0 H0 I0 G1 H1 I1 G2 H2 I2
D0 E0 F0 D1 E1 F1 D2 E2 F2
A0 B0 C0 A1 B1 C1 A2 B2 C2
R
$D_{min} = 3\sqrt{3}R$

Fig. 1. An optimal partition.

if a wireless channel can be allocated for supporting the communication between the MH and the MSS. Two cells can use the same channel only when the geographic distance between them is no less than 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 Fig. 1, R is the cell radius, $D_{\min}$ is the minimum channel reuse distance. If a fourth-power law attenuation is assumed [1], [16], 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. With $D_{\min} = 3\sqrt{3}R$, the channel reuse factor $N = (1/3)(D_{\min}/R)^2 = 9$. From Fig. 1, we can see that the channels are divided into nine groups (from A to I). Practical values [10] of N range from 3 ($D_{\min}/R = 3$ cell radii) to 21 ($D_{\min}/R = 7.9$). For example, in Advanced Mobile Phone Systems (AMPS) [6], $N = 12$ when omnidirectional antennas are used; $N = 7$ when $120°$ directional antennas are used. In *IS-136* (North American Cellular System Based on Time Division Multiple Access), $N = 7$ when the system is using $120°$ directional antennas. To simplify the presentation (especially the figure and the resource planning), we choose $N = 9$. Also, $N = 9$ is in the practical value range, and has been used by many previous works [1], [8].

1) Resource Planning Model: Most channel selection strategies require *a priori* knowledge of the channel status in order to achieve better channel reuse. For instance, in the channel borrowing strategies [9], [13], [17], 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 as *resource planning* [7], [8]. The following is a resource planning strategy which has three rules.

1) Partition the set of all cells into a number of disjoint subsets, $G_0, G_1, \ldots, G_{k-1}$, 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: $P_0, P_1, \ldots, P_{k-1}$.

2) The channels in $P_i (i = 0, 1, \ldots, k-1)$ 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 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 relative to r.

In order to achieve better channel reuse, each subset G_i should contain as many cells as possible. Then, k should be as small as possible. How to partition the cells is orthogonal to our discussion, but we require that the partition satisfies the following two properties, which have been proved to be the necessary conditions for any *optimal partition method* in [7] and [8].

Property 1:

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

Property 2:

$$\forall C_i, C_j: C_i \in IN_j \Rightarrow \forall r (IP_i(r) \cap IP_j(r) \neq \phi).$$

Property 1 is obvious. Property 2 is explained as follows: assume a cell C_i is an interference neighbor of C_j; if C_i and C_j request the same channel r, they have at least one common cell which is an interference primary cell of r.

Fig. 1 shows one partition, which divides the cells into nine subsets $G_A, G_B, \ldots, G_I$. Cells in $G_A = \{C_{A_i} | 0 \leq i \leq 8\}$ can use the same channel without interference. If two interference neighbors C_{G_2} and C_{D_5} request a primary channel of a cell in G_A, they have a common interference primary cell C_{A_5}. 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.

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 destination cell. This process is known as *handoff* (or *interhandoff*) [16]. During a handoff, the MH releases the currently used channel and is assigned a new channel by the destination MSS.

To achieve better channel reuse, *intrahandoff* (or a channel switch) may be necessary [2], [8]. In an intrahandoff operation, an MH releases its currently used channel and is assigned a new channel within the same cell. The motivation behind intrahandoff can be explained 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_f. 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_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 intrahandoff is of course the overhead. Fortunately, most of the channel selection strategies do not require a large number of intrahandoffs [2], [8]. Thus, intrahandoffs may be necessary for better channel reuse.

III. A Fault-Tolerant Channel Acquisition Algorithm

In the proposed channel acquisition algorithm, the borrower communicates with its interference neighbors to borrow a channel. Based on Property 2, when any two interference cells request the same channel r, they have at least one common cell which is an interference primary cell of r. Note that an interference primary cell of r is one element of the interference partition subset relative to r. If a cell gets permissions from all cells in an interference partition subset relative to channel r, it can borrow channel r without interfering other cells. Since the borrower only needs to receive responses from all cells in an interference partition subset rather than from all interference neighbors, the algorithm is fault-tolerant.

A. The Channel Acquisition Algorithm

In our distributed fault-tolerant channel acquisition algorithm, when a cell C_i needs a channel, it does the following. If it has an available primary channel r, it marks r as a *used channel* and uses r immediately; otherwise, C_i changes to *search mode* and sends a *request* message to each cell in IN_i. When a cell C_j receives the *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 [15] than C_i's, C_j sends a *reply* message which appends the information about its used channels to C_i; otherwise, C_j defers the *reply* (similar to [20]). After the borrower has received *reply* messages from cells in IN_i or the timer timeouts, the borrower computes the available channels and picks one according to the underlying channel selection algorithm. The borrower has to confirm the selected channel with the lenders, since a lender may assign that channel to a new call immediately after it sends out a *reply* message. If each lender responds with an *agree* message, the borrower can use the borrowed channel; otherwise, it picks another available channel and confirms again. If there is no available channel left, the call request is failed. After a channel has been borrowed, the lender marks the channel as an *interference channel*, and will not use it until it is returned by all borrowers.

A formal description of the algorithm is given in Fig. 2. Two types of control messages are used to acquire the information on available channels: *request* and *reply*. If two or more cells in each other's interference neighbors request the same channel concurrently, a conflict arises. If there is no conflict, three types of messages are exchanged among MSS's to confirm or return a channel: *confirm*, *agree*, and *release*. If there is a conflict, two additional messages are needed: *abort* and *conditional_agree*.

1) Conflict Resolution: In the proposed algorithm, control messages are timestamped using Lamport's clock [15] to determine the request priorities. The solution to conflicts is

<u>Data Structures</u>

- IN_i, $IP_i(r)$: defined before.
- B_i: a set of channels that C_i can borrow.
- S_i: a set of channels that C_i attaches with its *reply*.
- P_i: the set of primary channels assigned to C_i.
- U_i: the set of used channels at C_i. U_i is initialized to empty.
- $I_i(r)$: the set of cells to which C_i has sent an $agree(r)$ message. 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. $I_i(r)$ is initialized to empty.
- $CI_i(r, j)$: a set of cells, which saves the state of $I_i(r)$ when C_i receives C_j's *request* message.

(A) When a cell C_i needs a channel to support a call request, let $temp_i = P_i - U_i - \{r \mid r \in P_i \wedge I_i(r) \neq \phi\}$, if $temp_i$ is empty, C_i sets a timer and sends a *request* message to every cell $C_j \in IN_i$; otherwise, it picks a channel $r \in temp_i$ and adds r to U_i. When the call is finished, it deletes r from U_i.

(B) When a cell C_i receives a *request* from C_j, it sends $reply(P_i - U_i - \{r \mid r \in P_i \wedge (I_i(r) \cap IN_j \neq \phi)\})$ to C_j. For any $r \in P_i$, $CI_i(r, j) = I_i(r)$;

(C) When a cell C_i receives all $reply(S_j)$ messages or time out, it sets a new timer and does the following:

(C.1) for each r in the system, $B_i = B_i \cup r$ if
- C_i is not using r,
- for any $C_k \in IP_i(r)$, C_i has received a $reply(S_k)$ from C_k, and
- for any $C_j \in IP_i(r)$, $r \in S_j$

(C.2) if B_i is not empty, C_i selects a channel $r \in B_i$ using the underlying channel selection strategy and sends $confirm(r)$ messages to all cells in $IP_i(r)$; otherwise, it drops the call.

(D) When a cell C_i receives a $confirm(r)$ from C_j, its response depends on the current status of r:

(D.1) If $r \in P_i - U_i - \{r \mid r \in P_i \wedge I_i(r) \neq \phi\}$, C_i replies with an $agree(r)$ and adds C_j to $I_i(r)$.

(D.2) If $r \in U_i$, C_i replies with a $reject(r)$.

(D.3) If $I_i(r) \neq \phi$, let $temp_i = I_i(r) - CI_i(r, j)$, C_i does the following:

```
if $temp_i \cap IN_j = \phi$
then $C_i$ replies with an $agree(r)$ and adds $C_j$ to $I_i(r)$;
else if $\forall C_k \in (temp_i \cap IN_j)$, $C_j$'s request has a smaller timestamp than $C_k$'s request
    then $C_i$ replies with a $conditional\_agree(temp_i \cap IN_j, r)$ and adds $C_j$ to $I_i(r)$;
    else $C_i$ replies with a $reject(r)$.
```

(E) If C_i receives a response to its $confirm(r)$ from every cell in $IP_i(r)$ and the following two conditions are satisfied, then the request is successful:

(E.1) Every response is either an agree or conditional-agree.

(E.2) For each $conditional\_agree(S, r)$ and for each $C_j \in S$, C_i has received an $agree(r)$ from some $C_k \in (IP_i(r) \cap IP_j(r))$.

Otherwise, the request is failed. In case of success, C_i uses r to support the call and sends $release(r)$ to every cell in $IP_i(r)$ when the call terminates; in case of failure or time out, C_i sends an $abort(r)$ to those cells in $IP_i(r)$ from which it has received an agree or conditional-agree message, deletes r from B_i, and then goes to Step (C.2).

(F) Outdated *reply*, *agree*, and *conditional_agree* messages are discarded by comparing timestamps. When a cell C_i receives a $release(r)$ or $abort(r)$ from C_j, it deletes C_j from $I_i(r)$.

Fig. 2. The channel acquisition algorithm.

shown in Step D.3. By maintaining $I_i(r)$ and $CI_i(r, j)$, a cell C_i never grants concurrent requests for the same channel from cells within $D_{\min}$. In other words, if two cells, which are separated by a distance less than $D_{\min}$, request the same channel, they will not receive *agree* messages from the same interference primary cell.

Besides conflict resolution, the adoption of *conditional_agree* messages can also avoid wasting available channels, which can be explained by an example. In Fig. 1, assume that C_{H_4} and C_{F_4} concurrently request channels. Suppose C_{H_4}'s request has smaller timestamp. If there is one common available channel r in C_{A_4}, C_{A_5}, C_{A_7}, and C_{A_8}, C_{A_4}, C_{A_5}, C_{A_7}, and C_{A_8} are interference primary cells of r relative to C_{H_4} and C_{F_4}; i.e., $IP_{H_4}(r) = IP_{F_4}(r) = \{C_{A_4}, C_{A_5}, C_{A_7}, C_{A_8}\}$. If C_{H_4}'s *request* arrives at C_{A_4}, C_{A_5}, and C_{A_7} earlier than C_{F_4}'s *request*, but arrives at C_{A_8} later than C_{F_4}'s *request*, C_{A_4}, C_{A_5}, C_{A_7}, or C_{A_8} cannot send *agree* messages to both C_{H_4} and C_{F_4} due to the possibility of an interference. Without the use of *conditional_agree* messages, both C_{H_4} and C_{F_4} cannot use channel r. With the help of *conditional_agree* messages, after C_{H_4} gets *agree* messages from C_{A_4}, C_{A_5}, and C_{A_7}, and a *conditional_agree* from C_{A_8}, it can acquire r. Note that C_{F_4} cannot acquire r since C_{A_8} rejects its *request*.

2) Fault-Tolerance: In the proposed algorithm, a borrower does not need to receive a response from every interference neighbor. It only needs to receive a response from each cell in an interference partition subset as long as there is one common available channel among them. Based on Property 2, any two interference cells have at least one common interference primary cell of a channel. Also, the common interference primary cell never replies an *agree* message to more than one cell requesting the same channel. Thus, channel interference is avoided.

Since the number of cells in an interference partition subset is far less than the number of interference neighbors, our algorithm tolerates node failures and communication link failures. For example, in a typical cellular network model with $D_{\min} = 3\sqrt{3}R$, the number of interference neighbors of a cell is 30, and the number of interference primary neighbors of a cell is 3 or 4. In the best case, a cell can still borrow channels even though it cannot communicate with as many as $(30 - 3) = 27$ (i.e., $27/30 = 90\%$) of its interference neighbors. In the worst case, even though a cell cannot communicate with as many as $\lfloor(30/4)\rfloor = 7$ (i.e., $7/30 = 23\%$) of its interference neighbors, it can still communicate with the remaining $30 - 7 = 23$ cells, which includes all cells (at most 4) of an interference partition subset. If there are common available channels among cells in this interference partition subset, the cell can borrow these available channels.

3) Outdated Messages: Due to communication link failure or network congestion, messages such as *reply*, *agree*, and *conditional_agree* may arrive at a cell after the cell has finished the channel acquisition process. We call these messages *outdated messages*. Outdated messages must be identified and discarded; otherwise, two cells separated by a distance less than $D_{\min}$ may use the same channel, and then interfere with each other. In order to identify outdated messages, when a cell receives a message, such as *reply*, *agree* or *conditional_agree*, it compares the timestamp of the received message with that of its own *request* message. If the received message has a small timestamp compared to its own *request*, it is an outdated message. If a cell is not in the process of channel acquisition, all received *reply*, *agree*, and *conditional_agree* are outdated messages.

4) The Timer: Timers are used in our algorithm to deal with MSS or communication link failures. The selection of the timeout value affects the system performance. If the timeout value is too large, a handoff may be dropped due to the long delay. If the timeout period is too small, there may be less opportunity for the channel selection algorithm to choose a channel which can maximize channel reuse. The timeout value also depends on the applications. For example, a handoff request can tolerate much less delay than a new call request.

Suppose the time limit to borrow a channel is T_{limit}. For simplicity, we set the timer to $T_{\text{limit}}/2$. Under network congestion, communication link failures or MSS failures, the borrower waits $T_{\text{limit}}/2$ for receiving the *reply* messages, and another $T_{\text{limit}}/2$ for receiving the *confirm* responses. We do not set a different timeout value for the confirm responses due to the following reason. A borrower only sends the *confirm* messages to cells from which it has received the *reply* messages. Since the failure probability during this $T_{\text{limit}}/2$ time interval is very low, the borrower has a large probability to receive the *agree* messages from the lenders.

There are some other possible approaches. For example, there can be a timer for each round of confirm processes, so that the borrower can select another channel if a conflict or failure occurs during the confirming process. For simplicity, we use one time period in our algorithm.

5) The Acquisition Delay: The acquisition delay is the time required for an MSS to borrow a channel. In our algorithm, when some cells suffer from network congestion, a cell can still borrow channels from cells that are not experiencing network congestion, and hence our algorithm has low acquisition delay. Also, the acquisition delay deadline can be guaranteed by the timer. However, if a cell suffering from network congestion needs to borrow a channel, it is very likely to experience a long acquisition delay. Even in this situation, our algorithm still has shorter delay compared to nonfault-tolerant algorithms since our algorithm does not need to wait for the response messages from the cells suffering from network congestion.

Since the fault-tolerant algorithm needs two rounds of message exchanges to borrow a channel, the acquisition delay is $4*T$, where T is the one way communication delay. Recently, we proposed an efficient channel allocation algorithm [3] which can further reduce the channel acquisition delay to almost $2*T$. However, the algorithm [3] is not fault-tolerant. Moreover, the borrower requires to receive the *reply* message from each interference neighbor, and the acquisition delay should be twice the maximum communication delay from the borrower to any of its interference neighbor, denoted by $2*T_{\max}$. In the fault-tolerant algorithm, the borrower only needs to receive *reply* messages from each cell in an interference partition subset, and the acquisition delay should be four times the minimum communication delay from the borrower to cells in an interference partition subset, denoted by $4*T_{\min}$. Although $T_{\max}=T_{\min}$ most of the time, $T_{\max}\gg T_{\min}$ under network congestion. How to reduce the acquisition delay of our fault-tolerant channel allocation algorithm still needs further investigation. Since fault-tolerance is the major concern of this paper, we will not discuss the acquisition delay further.

B. Correctness Proof

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

Proof: Assume to the contrary that two cells C_i and C_j $(C_i\in IN_j)$ are using the same channel r at the same time. Since the distance between two primary cells is at least $D_{\min}$ (Property 1), C_i and C_j cannot both be the primary cells of the channel 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 $C_i\in IP_j(r)$. There are two possibilities.

Case 1.1: C_i receives C_j's *request* after its own request. To use r, C_i adds channel r to U_i (Step A). When C_j receives C_i's *reply* $(P_i-U_i-\{r|r\in P_i\wedge(I_i(r)\cap IN_j\neq\phi)\})$, $r\in U_i\implies r\notin B_j$ according to Step C.1. Then C_j cannot acquire r.

Case 1.2: C_i receives C_j's *request* before its own request. r is an interference channel when C_i starts the request (Step D.1), and C_i will not acquire r.

Case 2: C_j is a primary cell channel 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. According to Property 2, $IP_i(r)\cap IP_j(r)\neq\phi$. Without loss of generality, we assume that C_i's *request* has a small timestamp compared to C_j's *request*. If C_i finally acquires r, it must have received an *agree* message from a neighboring primary cell $C_k\in IP_i(r)\cap IP_j(r)$. There are two possibilities depending on when C_k receives C_j's *request*:

Case 3.1: C_k receives C_j's *request* after it sends *agree* to C_i. According to Step D.1 or D.3, C_k adds C_i to $I_k(r)$. When C_k receives C_j's *request*, $C_i\in I_k(r)\implies I_k(r)\cap IN_j\neq\phi$. Then, when C_j receives *reply* (B_k) from C_k, $r\notin B_k$ according to Step B. $r\notin B_k\implies r\notin B_j$ according to Step C.1. Thus, C_j cannot acquire r.

Case 3.2: C_k receives C_j's *request* before it sends *agree* to C_i. Then $C_i\notin CI_k(r,j)$. When C_k receives C_j's *confirm*, if $C_i\notin I_k(r)$, $C_j\in I_k(r)\implies temp_i\cap IN_i\neq\phi$, and then C_i can not get an *agree* from C_k (Step D.3). If $C_i\in I_k(r)$, we have $C_i\notin CI_k(r,j)\cap C_i\in I_k(r)\implies C_i\in(temp_k\cap IN_j)$. Since C_j's request has larger timestamp than C_i's request, S_k responses with a *reject* to C_j's *confirm* (Step D.3). Hence, C_j cannot acquire r. □

Theorem 2: The channel acquisition algorithm is deadlock free.

Proof: In the channel acquisition algorithm, an MSS receiving a *request* responds immediately by a *reply*. An MSS receiving a *confirm* also responds immediately by an *agree*, a *conditional_agree*, or a *reject*. A borrowing MSS uses a timer to make sure that it will not wait forever. Hence, our algorithm is deadlock free. □

C. Failure Recovery

Even though our channel acquisition algorithm is fault-tolerant, fast recovery techniques can significantly reduce the failure rate. Hence, we briefly describe how the algorithm can recover from failures.

1) MH Failures: When an MH fails in the middle of a communication session, the session is terminated. Hence, the channel that was being used for the communication session is no longer in use. The corresponding MSS detects the failure of the MH in its cell and deletes the channel from its U_i. Thus, as far as the channel allocation is concerned, the failure of an MH is conceptually handled in the same way as the completion of a communication session.

2) MSS Failures: We assume that MSS failures are fail-stop in nature. When an MSS, say C_i fails, all the communication sessions between C_i and MH's in its cell are terminated. Hence, no channel is in use inside C_i, after C_i fails and before it recovers. When C_i fails, its neighbors may still send *request*, *confirm*, *release*, or *abort* message to C_i. Since the borrower does not need to receive responses from all its interference neighbors, cells can still borrow channels which are not C_i's primary channels.

When C_i recovers from a failure, it needs to reconstruct its U_i and I_i as follows. C_i clears U_i and I_i, and broadcasts *cell_recovery* to all its interference neighbors. Any cell receiving *cell_recovery* replies with the channels that it borrowed from C_i. Based on these replies, C_i reconstructs its I_i. For example, adds C_j to $I_i(r)$ if C_j borrowed channel r from C_i. C_i can use sliding window protocols [21] to guarantee that every interference neighbor responds to its *cell_recovery* message.

3) Communication Link Failures: When there is a communication link failure, the underlying protocol such as the network layer should route messages through other links. Since the loss of control messages such as *release* and *abort* may significantly reduce system performance, sliding window protocols should be used to guarantee that the messages such as *abort* and *release* are not lost. In case of a network partition, a failure recovery procedure is used to recover from the loss of these two messages. During the recovery, an MSS, say C_i, broadcasts *link_recovery* to all its interference neighbors. A cell C_j receiving a *link_recovery* from C_i replies all the channels that C_j borrowed from C_i. When C_i receives these messages, it reconstructs its I_i similar to that in the MSS failure recovery.

Due to communication link failure or network congestion, messages such as *reply*, *agree*, and *conditional_agree* may arrive at a cell after the cell has terminated the channel acquisition process. How to deal with these outdated messages has been discussed in Section III-A.

IV. A Complete Channel Allocation Algorithm

As explained in early sections, a complete channel allocation algorithm includes a channel acquisition algorithm and a channel selection algorithm. We have presented a distributed channel acquisition algorithm in the last section. In this section, we provide a channel selection algorithm and integrate it into the channel acquisition algorithm to construct a complete channel allocation algorithm.

A. The Channel Selection Algorithm

Similar to the geometric strategy [1] and the channel selection algorithm in the update approach [8], our channel selection algorithm makes use of the optimal resource planning model defined in Section II, where 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 acquires the channel with the highest priority; when a cell releases a channel, it 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, where the used channel is released and the new available channel is assigned to the session supported by the released channel.

Our algorithm is different from the geometric and the update approaches when assigning priorities to the secondary channels. Similar to [23], 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, $G_0, G_1, \ldots, G_{k-1}$, as defined in Section II. Without loss of generality, we assume that there are a total of $k * n$ channels numbered $0, 1, \ldots, k*n-1$ which are evenly divided into k subsets: $P_0, P_1, \ldots, P_{k-1}$ (this assumption is not essential and is made only for ease of presentation). Let $P_l = \{l*n, l*n+1, \ldots, (l+1)*n-1\}, l = 0, 1, \ldots, k-1$.

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:

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

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

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

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

From (1), the primary channels in a cell have the highest priority since m is a significantly large number. For secondary channels, a 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 the higher number has the higher priority.

B. The Complete Channel Allocation Algorithm

Most of the existing DCS strategies [4], [5], [9], [12]–[14], [18], [22] need up-to-date information to calculate channel priority. This can be easily implemented in centralized algorithms, since the MSC monitors every release and acquisition of channels, and hence it has the 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 obtain the approximately up-to-date information by exchanging messages. For example, in the update approach [8], a cell sends *update notification* messages to its interference neighbors each time it acquires or releases a channel so that each cell always knows the available channels of its interference neighbors. In order to combine the channel selection algorithm with our distributed channel acquisition algorithm and do not significantly increase the message overhead, we make some modifications to our algorithm.

To make use of locality, a cell does not return the *borrowed* channel immediately after its use. Instead, it keeps the borrowed channel so that these channels can be used when the borrower runs out of channel again. Thus, there are two kinds of borrowed channels in the proposed algorithm: *used-borrowed channel* and *available-borrowed channel*. Used-borrowed channels are counted as used channels. Available-borrowed channels are counted as available channels and can be used again without contacting interference neighbors. When a cell finds that its lender's available channels are less than a threshold η (determined later), it releases the available-borrowed channels from that lender. It should perform an intrahandoff to release the used-borrowed channels from that lender if it is possible (it is not possible when there is no other available channel). Whenever a communication session in a cell is over, the cell checks if its available channels are larger than η'; if so, it releases a borrowed channel.

There are two approaches to reduce the update notification 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. As a result, cells keep the up-to-date information for calculating the channel priority of its interference neighbors. This approach significantly reduces the update notification 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 $\eta''(\eta'' > \eta)$.

The disadvantage of Approach 2 is that the borrower may not know the up-to-date information. The advantage of Approach 2 is low message overhead and low intrahandoff overhead. Knowing the up-to-date information is only helpful when releasing the borrowed channels. Since we want to make use of locality by keeping borrowed channels, and a borrowed channel will be released when its lender's available channels are less 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.

To make use of locality, η' should be as large as possible. However, keeping too many borrowed channels may increase the failure rate since other interference cells cannot use them. Certainly, we do not want to make use of locality at the expenses of increasing failure rate. Thus, we choose η' to be a small value. η should be as small as possible so that the borrower can keep the borrowed channel. However, if η is too small, the lender may run out of channel. η'' should be as small as possible to reduce the update notification message overhead. However, a larger value can help the borrower get more up-to-date information. Based on the above considerations and our simulation results, we choose η and η' to be 5% of the number of primary channels, η'' to be 10% of the number of primary channels. Since this is not the major concern of our paper, we will not further investigate how to choose the value of these parameters.

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. Certainly, if we choose the channel selection algorithm used in the update approach or the search approach, the update notification message overhead will be avoided. Note that the update notification messages are not required by the channel selection algorithm used in the update approach, but it is necessary for the channel acquisition algorithm used in the update approach.

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. 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 borrow the intrahandoffed channels.

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 failure rate.

V. Simulation Results

We evaluate the performance of the proposed channel allocation algorithm under two environments: without failures (of MSS's or communication links) and with failures. Without considering failures, we study the performance of the proposed channel allocation algorithm, the search approach [19], the update approach [8], and the geometric strategy [1] using extensive simulations. The performance of each strategy is simulated under both uniform and nonuniform traffic distributions.

When considering failures, we compare the performance of two channel allocation algorithms: one is the proposed fault-tolerant channel allocation algorithm, and the other is a nonfault-tolerant channel allocation algorithm which is a trivial modification of the proposed algorithm, where the borrower needs to consult with all interference neighbors. To avoid deadlocks in the modified nonfault-tolerant algorithm, a new communication session request is failed if the borrower cannot get all responses within a time limit. We do not compare our algorithm with other algorithms when considering MSS failures or communication link failures since all known distributed channel allocation algorithms *do not* provide fault tolerance.

A. Simulation Parameters

The simulated cellular network is a wrapped-around layout with $9 * 9$ 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. Under normal condition (no network congestion), 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.

TABLE I
SIMULATION PARAMETERS FOR UNIFORM TRAFFIC DISTRIBUTION

Mean arrival rate in a cell	λ
Mean inter-handoff rate in a normal cell	1/80s
Mean service time per communication session	180s

TABLE II
SIMULATION PARAMETERS FOR NONUNIFORM 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	1/80s
Mean inter-handoff rate in a hot cell	1/180s
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

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

Nonuniform traffic distribution is modeled by a two-state Markov Modulated Possion Process, where a cell can be in one of two states: *hot state* or *normal state*. As shown in Table II, 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 capture more arriving new users and prevailing stationary users. Each cell can dwell in either state for an exponentially distributed time independent of one another.

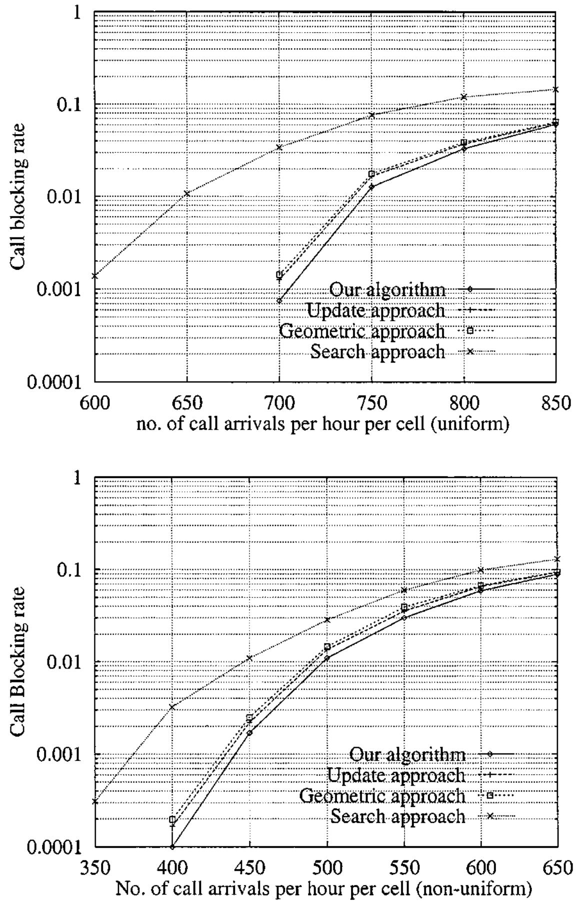


Fig. 3. Comparisons of failure rate.

B. Simulation Results (Without Failure)

For each call arrival rate, the mean value of the measured data 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.

The performance of the channel allocation algorithm is measured by the *failure rate* [1] $R_f = R_b + (1 - R_b) * R_d$, where R_b is the blocking rate and R_d is the dropping rate. The blocking rate is the percentage of new calls that are blocked due to the lack of an available channel, and the dropping rate is the percentage of ongoing calls that are dropped during interhandoffs because of insufficient resources.

1) A Comparison of Failure Rate: The failure 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 failure rate in these three approaches does not have too much difference, with our algorithm slightly outperforming the other two (see Fig. 3). 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 all of 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. This explains why our algorithm outperforms the update approach which outperforms the geometric strategy.

Compared to the search approach, our algorithm significantly reduces the failure rate. This is due to the fact that the search approach is a simple dynamic 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 intrahandoff since almost all advanced channel selection algorithms use intrahandoff to reduce the failure rate. The geometric strategy, the update approach, and our algorithm all use intrahandoff to achieve better channel reuse. The search approach does not have intrahandoff, but its failure rate is significantly higher than the other three. Note that when a call is blocked, the user may retry. In our simulation, the retry is counted as a call, and we do not differentiate between a retry and a new call.

2) Message Complexity per Channel Acquisition: As shown in Fig. 4, the number of messages per channel acquisition in the update approach [8] is never lower than $2 * n = 60$ (n is the number of interference neighbors), since a cell has to communicate with its interference neighbors whenever it acquires or releases a channel. In the search approach [19] and the proposed algorithm, a cell only communicates with its interference neighbors when it needs to borrow a channel. From Fig. 4, the message complexity of the search approach and the proposed

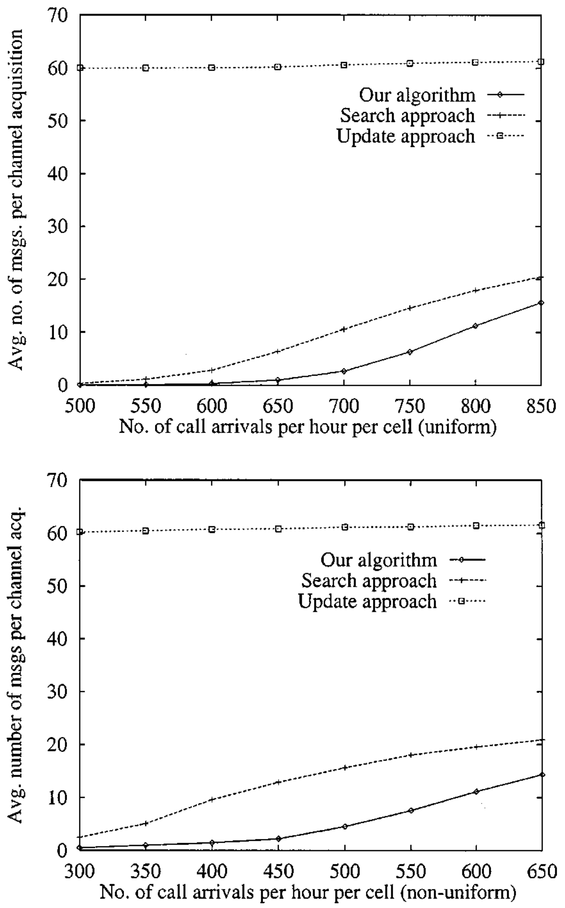


Fig. 4. Comparisons of message complexity.

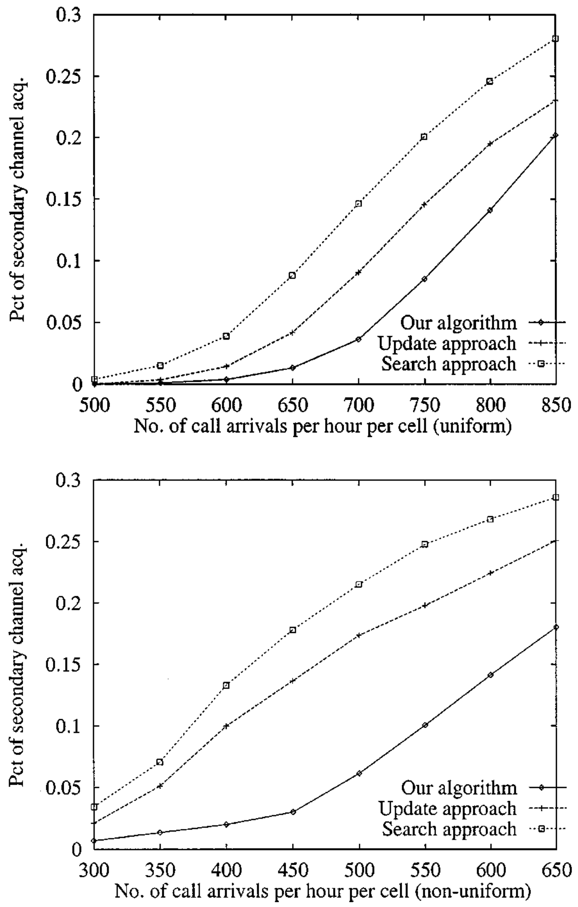


Fig. 5. Percentage of secondary channel acquisition.

algorithm increases from near 0 to about 20 as the channel request load increases. This can be explained by the fact that most of the call requests can be satisfied by the primary channel acquisition under low channel request load. As channel request load increases, more cells run out of primary channels and have to make more secondary channel acquisitions.

As shown in Fig. 4, although including update notification messages, our algorithm still has lower message complexity than the search approach under uniform traffic distribution. This can be explained by the fact that both algorithms have different secondary channel acquisition percentage. From Fig. 5, 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, a cell just randomly borrows a channel from its neighbors. 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. Also, keeping the borrowed channels reduces the number of channel borrowing. Note that acquiring an available-borrowed channel does not counted as a secondary channel acquisition, since in this case, the borrower does not need to contact its interference neighbors.

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. As a result, free channels are transferred to these hot cells, and 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 channel; when the traffic load is low, most of them have free channels. Thus, the advantage of keeping channel under uniform traffic distribution is not that significant compared to that under nonuniform traffic distribution. This explains why our approach has much lower secondary channel acquisition percentage than other approaches under the condition of nonuniform 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 after this point. Certainly, it is still lower than the secondary channel acquisition percentage in other approaches.

C. Simulation Results (With Failures)

In this subsection, we compare the performance of the fault-tolerant and nonfault-tolerant channel allocation algorithms under nonuniform traffic distribution.

1) The Failure Rate Under MSS Failures or Network Partitioning: If an MSS fails, every call requests in that cell fails.

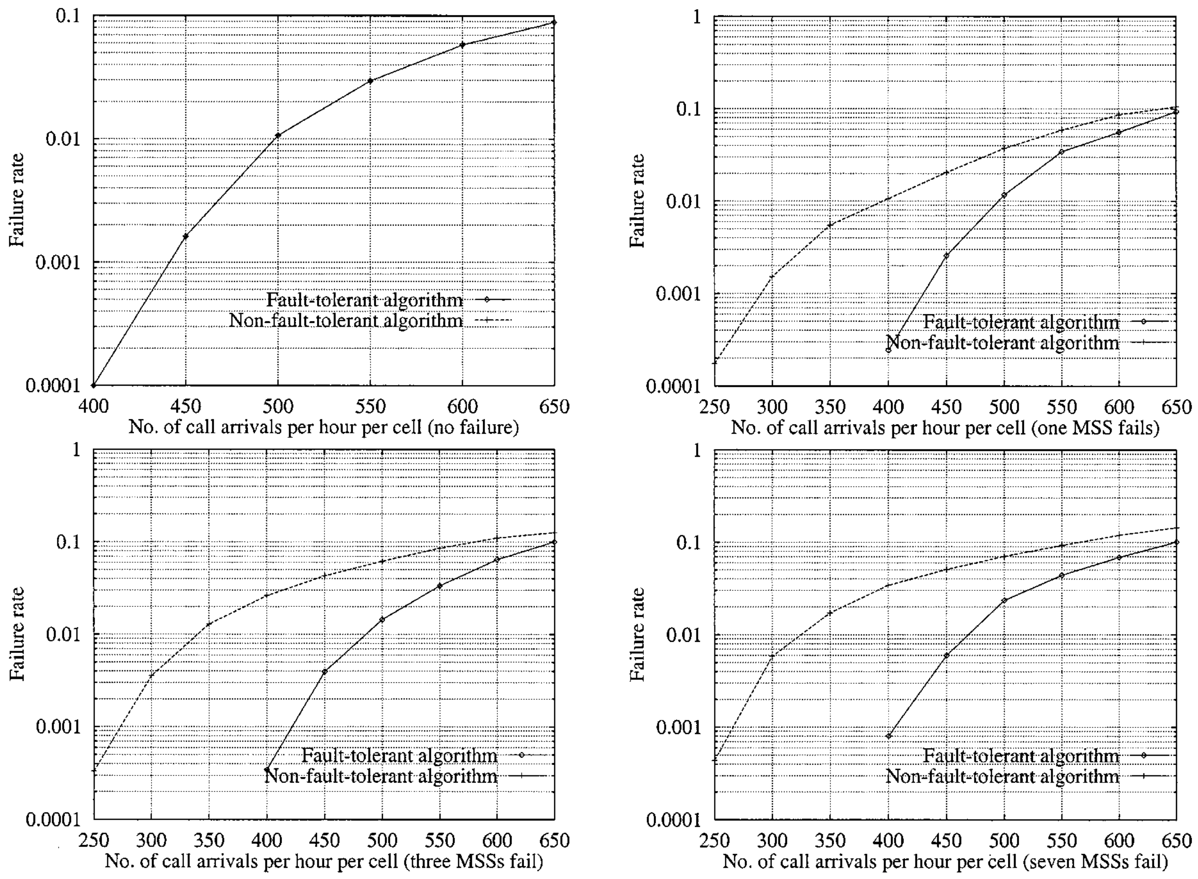


Fig. 6. Failure rate under MSS or communication link failures.

Even though the failure rate in the failed cell is 100%, other cells may still have a very low failure rate. To reflect the performance of operational cells, the failure rate is calculated without considering the failed cells when there are MSS failures. Note that an MSS failure *does not* necessary mean that the MSS has crashed. A failed MSS may still be able to support its MH's, but it cannot communicate with other MSS's, and then it cannot borrow (lend) channels from (to) other MSS's.

Fig. 6 compares the failure rate of the fault-tolerant algorithm and the nonfault-tolerant algorithm under four conditions: no failure, one MSS failure, three MSS failures, and seven MSS failures. As shown in Fig. 6, the failure rate of the nonfault-tolerant channel allocation algorithm is significantly higher (about 100 times when there are 400 call arrivals per hour per cell) than that of the fault-tolerant channel allocation algorithm. This can be explained by the follows. In the nonfault-tolerant channel allocation algorithm, a cell cannot borrow any channel from its neighbors if it cannot communicate with any of its interference neighbors. As a result, a cell can only use its primary channels when it cannot communicate with any of its 30 interference neighbors, and hence the nonfault-tolerant algorithm has a high failure rate under MSS failures. However, in the fault-tolerant channel allocation algorithm, a cell can still borrow channels even if it cannot communicate with some interference neighbors. For example, with $D_{\min} = 3\sqrt{3}R$, the number of interference neighbors of a cell is 30, and the number of interference primary neighbors of a cell is 3 or 4. In the best case, a cell can still borrow channels even though it cannot communicate with as many as $(30 - 3) = 27$ (i.e., $27/30 = 90\%$) of its interference neighbors. In the worst case, even though a cell cannot communicate with as many as $\lfloor(30/4)\rfloor = 7$ (i.e., $7/30 = 23\%$) of its interference neighbors, it can still communicate with the remaining $30 - 7 = 23$ cells, which includes all cells (at most 4) of an interference partition subset. If there are common available channels among cells in this interference partition subset, the cell can borrow these available channels, and hence, the fault-tolerant algorithm has a low failure rate compared to the nonfault-tolerant algorithm under MSS failures.

From Fig. 6, we can see that the failure rate with MSS failures is higher than the failure rate without MSS failures in the fault-tolerant channel allocation algorithm. This can be explained by the fact that the failed MSS may have borrowed some channels and these channels cannot be used by the lenders until the failed MSS is recovered. (The figure shows the failure rate without considering recovery.)

The fault-tolerant channel allocation algorithm exhibits the useful property of *graceful degradation*, which is highly desirable in distributed fault-tolerant computing systems. In a failure-free environment, the algorithm has low failure rate. As MSS failures occur and increase, the number of available channels decreases and the failure rate increases.

2) The Failure Rate Under Network Congestion: Network congestion depends on a large number of parameters, e.g., the number of arriving messages, the queue length, the service rate,

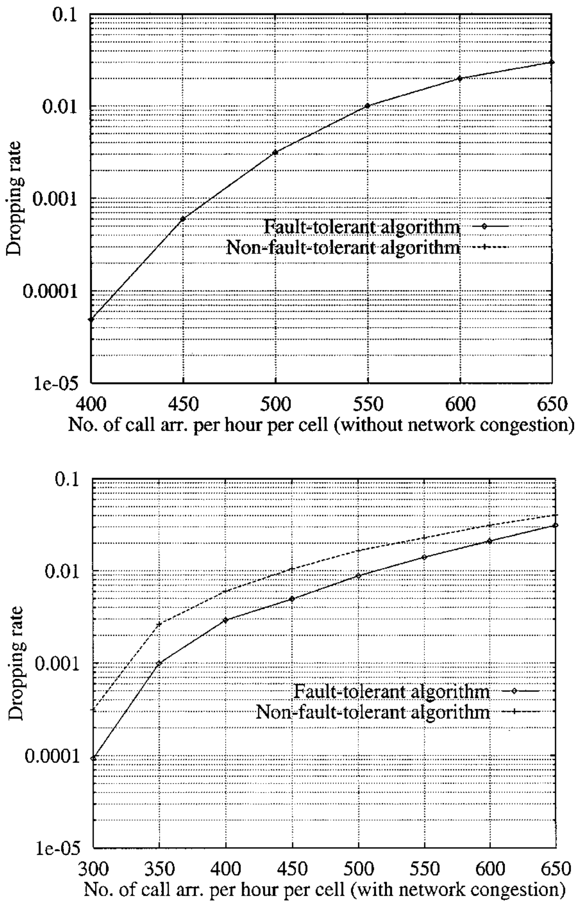


Fig. 7. Dropping rate under network congestion.

etc. Since an MSS has other functionalities besides handling channel borrowing, it is very difficult to model a congested network. To simplify the model, we assume that a network congestion only occurs in the hot cells. The network congestion has an exponentially distributed time with a mean time of 60 s. During a network congestion, the communication delay increases to 4 ms. The maximum tolerable delay of an interhandoff is much less than that of a call request. We assume the maximum tolerable delay of an interhandoff is 10 ms.

In the nonfault-tolerant algorithm, the borrower needs to wait for the reply from every interference neighbor, and it needs more than $4 * 4 = 16$ ms to borrow a channel. Since the waiting time (16 ms) is longer than the maximum interhandoff delay 10 ms, the handoff requests are dropped during network congestion in the nonfault-tolerant algorithm. However, in the fault-tolerant algorithm, a cell can still borrow a channel from other neighbors which does not suffer from network congestion. In this way, the borrower needs only $2 * 4 = 8$ ms <10 ms to borrow a channel. This explains the results of Fig. 7, where the fault-tolerant algorithm has lower handoff dropping rate than the nonfault-tolerant algorithm.

From Fig. 7, we can see that the fault-tolerant channel allocation algorithm under network congestion has much higher call dropping rate compared to the dropping rate without network congestion. This can be explained as follows. Network congestion occurs when a cell is in the hot state, which is reflected by low interhandoff rate. Suppose C_i is suffering from network congestion. It is more likely that most of C_i's neighbors are still in the normal state, which has high interhandoff rate. In other words, there are much more MH's coming from C_i's neighbors than those leaving C_i. Since C_i is experiencing network congestion, any communication with outside cells has long delay, which results in high dropping rate. Certainly, it is still much lower than the dropping rate in the nonfault-tolerant algorithm. Note that cells in both the fault-tolerant and nonfault-tolerant algorithms keep the borrowed channels for some time. This reduces the dropping rate when cells are in the hot state, but when a cell changes from the normal state to the hot state, the dropping rate is still high. Based on this reasoning, if a cell begins to borrow channels when the number of its available channels is below some threshold (a system tuning factor) instead of waiting for running out of channel, the dropping rate of the fault-tolerant algorithm may be further reduced. If a different model (i.e., network congestion occurs randomly) is used, the dropping rate of the fault-tolerant channel allocation algorithm will be much lower since the congested cell may not be in the hot state. However, the dropping rate of the nonfault-tolerant channel allocation algorithm will not change since the borrower needs to communicate with its congested interference neighbors even though they are in the hot state.

VI. Conclusion

Distributed channel allocation algorithms have received considerable attention because of their high reliability and scalability. However, previous algorithms cannot tolerate any communication link failures and node failures. Moreover, these algorithms have poor performance under network congestion, which may happen frequently under heavy traffic load.

In this paper, we proposed a fault-tolerant channel acquisition algorithm which tolerates communication link failures and node failures. In the algorithm, a borrower does not need to receive a response from every interference neighbor. It only needs to receive a response from each cell in an interference partition subset as long as there is one common available channel among them. Since the number of cells in an interference partition subset is far less than the number of interference neighbors, our algorithm tolerates network congestion, communication link failures, and node failures. Based on the typical cellular network model, in the best case, a cell can still borrow channels even though it cannot communicate with as many as 90% of its interference neighbors. In the worst case, even though a cell cannot communicate with as many as 23% of its interference neighbors, it can still borrow channels. Detailed simulation experiments were carried out in order to evaluate our proposed methodology. Simulation results showed that our algorithm significantly reduces failure rate under network congestion, communication link failures, and node crashes compared to nonfault-tolerant channel allocation algorithms. Moreover, the proposed fault-tolerant algorithm reduces the message overhead compared to known distributed channel allocation algorithms, and outperforms them in terms of failure rate under uniform as well as nonuniform traffic distribution.

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. Technol.*, vol. 44, pp. 14–23, Feb. 1995.

[2] R. Beck and H. Panzer, "Strategies for handover and dynamic channel allocation in micro-cellular mobile radio systems," *IEEE Trans. Veh. Technol.*, vol. 38, pp. 668–672, May 1989.

[3] G. Cao and M. Singhal, "An adaptive distributed channel allocation strategy for mobile cellular networks," *J. Parallel Distributed Computing, special issue on Mobile Computing*, vol. 60, no. 4, pp. 451–473, Apr. 2000.

[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)*, vol. 3, no. 5, pp. 333–347, 1997.

[5] ——, "A novel load balancing scheme for the tele-traffic hot spot problem in cellular network," *ACM/Baltzer Wireless Networks (WINET)*, vol. 4, no. 4, pp. 325–340, 1998.

[6] S. B. Davidson, H. Garcia-Molina, and D. Skeen, "Advanced mobile phone service: The cellular concept," *Bell System Tech. J.*, pp. 15–41, Jan. 1979.

[7] X. Dong and T. H. Lai, "An efficient priority-based dynamic channel allocation for mobile cellular networks," presented at the IEEE INFOCOM, 1997.

[8] ——, "Distributed dynamic carrier allocation in mobile cellular networks: Search vs. update," in *Proc. Int. Conf. Distributed Computing Syst.*, May 1997, pp. 108–115.

[9] S. M. Elnoubi, R. Singh, and S. C. Gupta, "A new frequency channel assignment algorithm in high capacity mobile communication systems," *IEEE Trans. Veh. Technol.*, vol. 31, pp. 125–131, Aug. 1982.

[10] D. Goodman, *Wireless Personal Communications Systems*. Reading, MA: Addison-Wesley, 1997.

[11] ——, "Cellular packet communication," *IEEE Trans. Commun.*, vol. 38, pp. 1272–1280, Aug. 1990.

[12] H. Jiang and S. Rappaport, "Prioritized channel borrowing without locking: A channel sharing strategy for cellular communications," *IEEE/ACM Trans. Networking*, vol. 4, pp. 163–172, Apr. 1996.

[13] T. J. Kahwa and N. D. Georganas, "A hybrid channel assignment scheme in large-scale, cellular-structured mobile communication systems," *IEEE Trans. Commun.*, vol. 26, pp. 432–438, Apr. 1978.

[14] I. Katzela and M. Naghshineh, "Channel assignment schemes for cellular mobile telecommunication systems: A comprehensive survey," *IEEE Personal Commun.*, vol. 3, pp. 10–31, June 1996.

[15] L. Lamport, "Time, clocks and ordering of events in distributed systems," *Commun. ACM*, vol. 21, no. 7, pp. 558–565, July 1978.

[16] V. H. Macdonald, "Advanced mobile phone service: The cellular concept," *Bell Syst. Tech. J.*, pp. 15–41, Jan. 1979.

[17] S. Nanda and D. J. Goodman, "Dynamic resource acquisition: Distributed carrier allocation for TDMA cellular systems," in *Proc. GLOBECOM'91*, Dec. 1991, pp. 883–889.

[18] L. Ortigozo-Guerrero and D. Lara-Rodriguez, "A compact pattern with maximized channel borrowing strategy for mobile cellular networks," in *Proc. IEEE Int. Symp. Personal, Indoor, Mobile Radio Commun.*, 1996, pp. 329–333.

[19] R. Prakash, N. Shivaratri, and M. Singhal, "Distributed dynamic channel allocation for mobile computing," in *Proc. 14th ACM Symp. Principles Distributed Computing*, 1995, pp. 47–56.

[20] G. Ricart and A. K. Agrawal, "An optimal algorithm for mutual exclusion in computer networks," *Commun. ACM*, vol. 24, no. 1, Jan. 1981.

[21] W. Stalling, *Data and Computer Communications*. Englewood Cliffs, NJ: Prentice-Hall, 1996.

[22] J. Tajima and K. Imamura, "A strategy for flexible channel assignment in mobile communication systems," *IEEE Trans. Veh. Technol.*, vol. 37, May 1988.

[23] M. Zhang and T. S. Yum, "Comparisons of channel assignment strategies in cellular mobile telephone systems," *IEEE Trans. Veh. Technol.*, vol. 38, pp. 211–215, Nov. 1989.

Guohong Cao (S'98–A'99) received the B.S. degree from Xian Jiaotong University, Xian, China; and the M.S. and Ph.D. degrees in computer science from the Ohio State University in 1997 and 1999, respectively.

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.

Dr. Cao was a recipient of a Presidential Fellowship at The Ohio State University.

Mukesh Singhal (A'92–SM'98) received the Bachelor of Engineering degree in electronics and communication engineering (with high distinction) from the University of Roorkee, Roorkee, India, in 1980, and the Ph.D. degree in computer science from the University of Maryland, College Park, in May 1986.

He is a Full Professor of computer and information science at The Ohio State University, Columbus. His current research interests include operating systems, database systems, distributed systems, performance modeling, mobile computing, and computer security. He has published more than 130 refereed articles in these areas. He has coauthored two books titled *Advanced Concepts in Operating Systems* (New York: McGraw-Hill, 1994) and *Readings in Distributed Computing Systems* (IEEE Computer Society Press, 1993). He is currently the Program Director of Operating Systems and Compilers program at the National Science Foundation.