Distributed Fault-Tolerant Channel Allocation for Mobile Cellular Networks

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

Abstract

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 because they depend on a *Mobile Switching Center (MSC)* to accomplish channel acquisition, which are referred to as *centralized* channel acquisition algorithms. 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, a borrower fails to borrow channels when it cannot communicate with anyone of its interference neighbors. 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 these 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 non-fault-tolerant channel allocation algorithms. Moreover, our algorithm has lower message overhead compared to known distributed channel allocation algorithms, and outperforms them in terms of failure rate under uniform as well as non-uniform traffic distribution.

Key words: Distributed channel allocation, fault-tolerant, cellular network.

1 Introduction

Cellular communication networks divide the geographical area into smaller regions, called cells [8]. 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 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, 14], 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 channel available for use when the call is being set up or when it is being handed over to another cell.

Channel selection algorithms have been an active research topic [3, 7, 8, 9, 10, 11, 15, 19]. However, most of these algorithms depend on a *Mobile Switching Center (MSC)* to accomplish channel acquisition, which are referred to as *centralized* channel acquisition algorithms. More specifically, each cell notifies the MSC when it acquires or releases a channel so that the MSC knows which channels are available in each cell at any time and assigns channels to cells accordingly. Obviously, the centralized approach has low reliability and scalability.

Recently, distributed channel acquisition algorithms [6, 16] have received considerable attention because of their high reliability and scalability. In this approach, an MSS communicates with other MSSs directly to find the available channels and make sure that the channel assignment does not interfere with other cells. In general, there are two approaches in distributed channel acquisition algorithms: *Search* [16] and *Update* [6]. In the search approach [16], 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 [6], 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.

Both approaches require that a borrower waits for the acknowledgment 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 will have the same effect as a communication link failure or node (MH or MSS) failure, where a borrower may fail to borrow a channel even though there exists available channels in its neighbors. 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, 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 are not 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, a borrower does not need to receive a response from every interference neighbor. It only needs to receive a response 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 a channel from them. We also present a channel selection algorithm and integrate it into the distributed 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 non-fault-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 non-uniform traffic distribution.

The rest of the paper is organized as follows. Section 2 presents the system model. In Section 3, we propose a fault-tolerant channel acquisition algorithm, give correctness proofs, and propose some recovery strategies. Section 4 presents a channel selection algorithm and integrates it into the channel acquisition algorithm. Simulation results are presented in Section 5. Section 6 concludes the paper.

2 System Model

The geographical area is divided into hexagonal cells in a mobile cellular network [8]. Each cell is supported by an MSS in the center (we use cell and MSS interchangeably). The MSSs 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 [4], 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 supported if a wireless channel can be allocated for communication between the MH and the MSS. Two cells can use the same channel if the geographic distance between them is 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 Figure 1, R is the cell radius, D_{min} is the minimum channel reuse distance. If a fourth-power law attenuation is assumed [1, 13], the signal to interference ratio is given by $[S/I]_{min} = [(D_{min}/R) - 1]^4/6$. With $D_{min} = 3\sqrt{3}R$, $[S/I]_{min} \approx 17dB$, which is a reasonable value in practice.

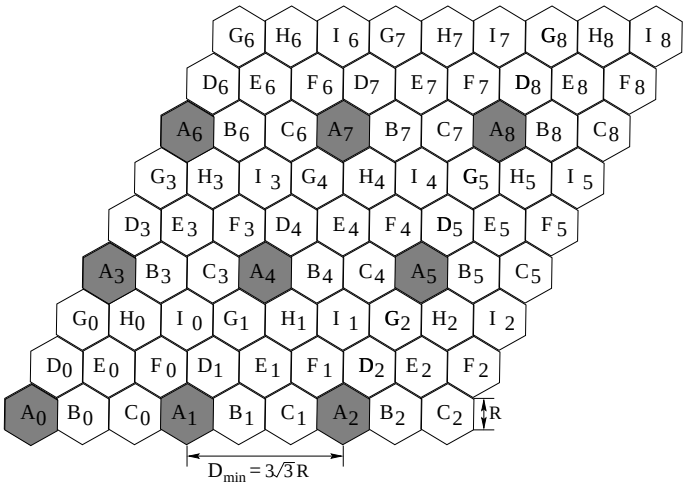

Figure 1: An optimal partition

Resource Planning Model

Most channel selection strategies require a *priori* information about channel status in order to

achieve better channel reuse. For instance, in the channel borrowing strategies [7, 10, 14], 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* [5, 6].

Resource Planning

1. Partition the set of all cells into a number of disjoint subsets, $G_0, G_1, \cdots, G_{k-1}$, such that any two cells in the same subset are apart by at least a distance of D_{min}. Accordingly, partition the set of all channels into k disjoint subsets: $P_0, P_1, \cdots, P_{k-1}$.
2. The channels in $P_i (i = 0, 1, \cdots, k-1)$ are the *primary channels* of cells in G_i, and the *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 partitions satisfy the following two properties, which have been proved to be the necessary conditions for any *optimal partition method* in [5, 6]:

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.

Figure 1 shows one partition, which divides the cells into nine subsets $G_A, G_B, \cdots, G_I$. Cells in $G_A = \{C_{A_i} | 0 \leq i \leq 8\}$ can use the same channel without interference. 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.

Handoff and intra-Handoff

A mobile host may cross the boundary between two cells while being active. When this occurs, the necessary state information must be transferred from its previous MSS to the MSS in the new cell. This process is known as *handoff* (or *inter-handoff*) [13]. During a handoff, an MH releases its current channel to its previous MSS and is assigned a new channel by the new MSS.

To achieve better channel reuse, *intra-handoff* (or a channel switch) may be necessary [2, 6]. In an intra-handoff operation, an MH releases its current channel and is assigned a new channel within the same cell. The motivation behind intra-handoff can be understood by an example. In Figure 1, suppose cell C_{F_1} borrows a channel $r1$ from A_1 and assigns it to a mobile host MH_f. Then, cells C_{A_1}, C_{A_2}, C_{A_4}, and C_{A_5} cannot use channel $r1$ due to interference. If a call in C_{F_1} terminates and a primary channel $r2$ is released, an intra-handoff from $r1$ to $r2$ by MH_f improves channel reuse, since $r1$ can be reused by four other cells C_{A_1}, C_{A_2}, C_{A_4}, and C_{A_5}. A drawback of intra-handoff is of course the overhead. Fortunately, most of the channel selection strategies do not demand many intra-handoffs [2, 6]. Thus, intra-handoff may be necessary for better channel reuse.

3 A Fault-Tolerant Channel Acquisition Algorithm

In the proposed algorithm, a 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 a cell of the interference partition subset relative to r. Thus, if a cell gets permissions from all cells in an interference partition subset relative to channel r, it can borrow channel r. As a result, a borrower only needs to receive responses from all cells in an interference partition subset rather than of all interference neighbors, which makes the algorithm fault-tolerant.

3.1 The Channel Acquisition Algorithm

In our distributed fault-tolerant channel acquisition algorithm, when a cell C_i needs a channel, if it has an available primary channel r, it marks r as an *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 [12] 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 [17]). When

C_i has received *reply* messages from cells in IN_i or timeout, it computes the available channels and picks one according to the underlying channel selection algorithm. The borrower C_i 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, C_i 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. When a channel r is borrowed, the lender marks r as an *interference channel*, and it can not use r until r is returned by all borrowers.

A formal description of the algorithm is given in Figure 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 for the same channel concurrently, a conflict arises. If there is no conflict, three types of messages are exchanged among MSSs to confirm or return a channel: *confirm*, *agree*, and *release*. If there is a conflict, two additional messages are needed: *abort* and *conditional_agree*.

Conflict Resolution: In our algorithm, control messages are timestamped using Lamport's clock [12] to determine the priority of requests. The solution to conflicts is 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}. Hence, no two cells requesting the same channel within D_{min} receive *agree* messages from the same interference primary cell.

Besides conflict resolution, the adoption of *conditional_agree* messages can also avoid wasting available channels as follows. In Figure 1, assume C_{H_4} and C_{F_4} concurrently request a channel, and 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}, then $C_{A_4}, C_{A_5}, C_{A_7}$, 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. Then, without *conditional_agree* messages, both C_{H_4} and C_{F_4} cannot use channel r. With *conditional_agree* messages, 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}, then it can acquire r. However, C_{F_4} cannot acquire r since C_{A_8} rejects its *request*.

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

Data Structures

- 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 $I_i(r)$.

Figure 2: The channel acquisition algorithm

interference neighbors, our algorithm tolerates node 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. Then, 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.

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 terminated the channel acquisition process. We call these messages *outdated messages*. Outdated messages must be identified and discarded; otherwise, two cells may 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 the timestamp of its own *request* message. If the received message has a smaller timestamp than its own *request*, it is an outdated messages. Also, if a cell is not in the process of channel acquisition, all received *reply*, *agree*, and *conditional_agree* are outdated messages.

The Timer: Timers are used in our algorithm to deal with MSS or communication link failures. The selection of the timeout period affects the system performance. If the timeout period is too long, a handoff may be dropped due to the long delay. If the timeout period is too short, there may be less opportunity for the channel selection algorithm to choose a channel. The timeout period also depends on the application. 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 $T_{limit}/2$. Then, under network congestion, link failures or MSS failures, the borrower waits $T_{limit}/2$ time interval for *reply* messages, and another $T_{limit}/2$ time interval for $confirm$ responses. We do not set a different timeout period for confirm responses. The reason is as follows. A borrower only sends $confirm$ messages to cells from which it has received *reply* messages. Since the probability that a failure occurs during this $T_{limit}/2$ time interval is very low; it has a large probability to receive an *agree* message from them.

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

Real-Time Communication: Our algorithm can *partially* support real-time communication.

Even when some cells suffer from network congestion, a cell can still borrow channels from cells that are not experiencing network congestion. Also, a delay deadline is 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 delay. Even in this situation, our algorithm still has shorter delay compared to non-fault-tolerant algorithms since our algorithm does not need to wait for the response messages from the cells suffering from network congestion.

3.2 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. 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 channel 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 channel r, C_i adds r to U_i (Step A). When C_j receives C_i's $reply(P_i - U_i - \{r \mid r \in P_i \wedge (I_i(r) \cap IN_j \neq \phi)\})$, $r \in U_i \Longrightarrow r \notin B_j$ 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), then C_i will not acquire r.

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. 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 smaller timestamp than C_j's *request*. If C_i finally acquires r, then 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*, $C_i \in I_k(r) \Longrightarrow I_k(r) \cap IN_j \neq \phi$. Then, when C_j receives $reply(B_k)$ from C_k, $r \notin B_k$ according to Step B. $r \notin B_k \Longrightarrow r \notin B_j$ 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 $C_i \notin I_k(r)$, $C_j \in I_k(r) \Longrightarrow temp_i \cap IN_i \neq \phi$, and then C_i can not get an *agree* from C_k (Step D.3). If $C_i \in I_k(r)$, we have $C_i \notin CI_k(r, j) \cap C_i \in I_k(r) \Longrightarrow C_i \in (temp_k \cap IN_j)$. 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. □

3.3 Failure Recovery

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

***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 channel allocation is concerned, failure of an MH is conceptually handled in the same way as the completion of a communication session.

***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 MHs in its cell are terminated. Hence, no channel is in use inside C_i after C_i fails before it recovers. When C_i fails, its neighbors may still send *request*, $confirm$, *release*, or *abort* message to C_i. Since a cell does not need to receive responses from all its interference neighbors, other cells can still borrow channels that 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 then it 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 protocol [18] to make sure that every interference neighbor responds to its *cell_recovery* message.

Communication Link Failures: When there is a communication link failure, the underlying protocol at the network layer should route messages from some other links. Since the loss of control messages such as *release* and *abort* may significantly reduce system performance, sliding window protocol 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 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 3.1.

4 A Complete Channel Allocation Algorithm

In this section, we provide a channel selection algorithm and integrate it into the channel acquisition algorithm.

4.1 The Channel Selection Algorithm

Similar to the Geometric strategy [1] and the channel selection algorithm in the update approach [6], our channel selection algorithm makes use of the optimal resource planning model defined in Section 2. The primary channels for each cell are prioritized. During a channel acquisition, a cell acquires the available primary channel that has the highest priority. If none of the primary channels is available, the cell borrows a channel from its neighbors according to some priority assignment approach. When a cell acquires a channel, it always acquires the channel with the highest priority. When a cell releases a channel, it always releases the channel with the lowest priority. If a newly available channel has a higher priority than some used channel, an intra-handoff is performed.

Our algorithm is different from the Geometric and the update approaches when assigning priority to secondary channels. Similar to [20], in our algorithm, a cell borrows the channel that has the lowest priority from the "richest" interference neighbors; i.e., the cell with the most available primary channels. The motivation behind this is to reduce the chance that the lender might soon use up its primary channels and have to acquire a secondary channel. In the following, we formally define the channel priority used in our channel selection algorithm.

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

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

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

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

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

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

From Equation (1), the primary channels in a cell have the highest priority since m is a 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.

4.2 The Complete Channel Allocation Algorithm

In order to combine the channel selection algorithm with our channel borrowing algorithm, we modify our channel acquisition algorithm as follows: when a primary channel becomes available or not available (a channel is not available if it is used or interference with others), it notifies all cells which have borrowed channels from it. More formally, whenever a cell C_i deletes (adds) $r \in P_i$ from (to) U_i, or $I_i(r)$ changes from empty (non-empty) to non-empty (empty), it sends $update(r)$ to each cell in $\{C_j \mid r' \in P_i \wedge C_j \in I_i(r')\}$. When a cell C_i receives an $update(r)$ from C_j, it updates the local information about C_j, and then it keeps the up-to-date information for calculating the "richness" of its interference neighbors.

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

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

5 Simulation Results

We evaluate the performance of the proposed channel allocation algorithm under two environments: without failure (of MSSs or communication links) and with failures. Without considering failures, we study the performance of the proposed channel allocation algorithm, the Search approach [16], the Update approach [6], and the Geometric strategy [1] using extensive simulations. The performance of each strategy is simulated under both uniform and non-uniform traffic distributions.

When considering failures, we compare the performance of two channel allocation algorithms: one is the proposed fault-tolerance channel allocation algorithm, and the other is a non-fault-tolerant channel allocation algorithm which is a trivial modification of the proposed algorithm, where a borrower needs to consult with all interference neighbors. To avoid deadlocks in the non-fault-tolerant algorithm, a call request fails if the borrower can not get all responses within a time limit. We do not compare our algorithm with other algorithms under failure environment since all known distributed channel allocation algorithms *do not* provide fault tolerance.

5.1 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 MSSs is 2 milliseconds, which covers the transmission delay, the propagation delay, and the message processing time.

Under uniform traffic distribution (shown in Table 1), 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.

Non-uniform 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 2, a cell spends most of its time in normal state. A cell in normal state is characterized by low arrival rate and high inter-handoff rate. On the contrary, a cell in hot state is characterized by high arrival rate and low inter-handoff rate to 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.

Table 1: **Simulation Parameters for uniform traffic distribution**

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

5.2 Simulation Results (Without Failure)

For each arrival rate, the mean value of a 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 =

Table 2: **Simulation parameters for non-uniform traffic distribution**

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

$R_b + (1 - R_b) * R_d$, 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 insufficient resources, and the dropping rate is the percentage of on-going calls that are dropped during inter-cell handoffs due to insufficient resources.

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

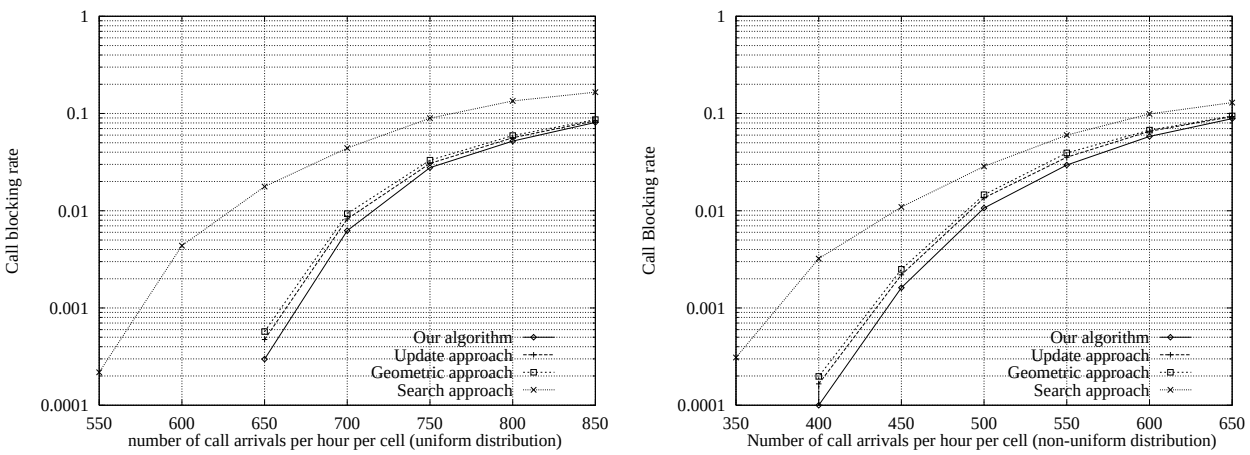

Figure 3: Comparisons of failure rate

5.2.2 Message Complexity Per Channel Acquisition

As shown in Figure 4, the number of messages per channel acquisition in the update approach [6] is never lower than $2 * n = 60$, since a cell has to communicate with its interference neighbors whenever it acquires or releases a channel (even when it acquires or releases a primary channel). In the search approach [16] and the proposed algorithm, a cell only communicates with its interference neighbors when it needs to borrow a channel. From Figure 4, the message complexity of the search approach and the proposed 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.

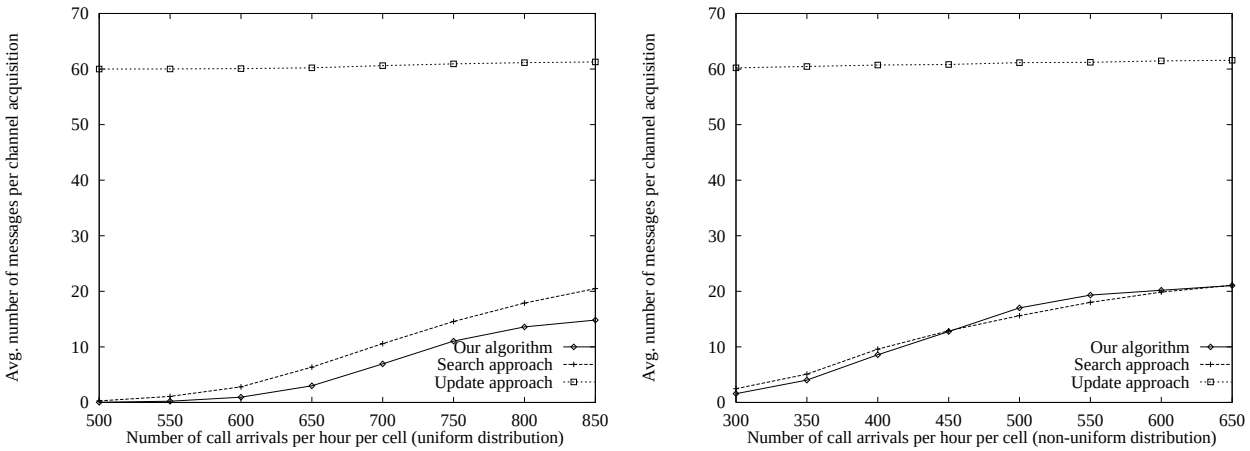

Figure 4: Comparisons of message complexity

As shown in Figure 4, although including update notification messages (shown in Figure 6), our

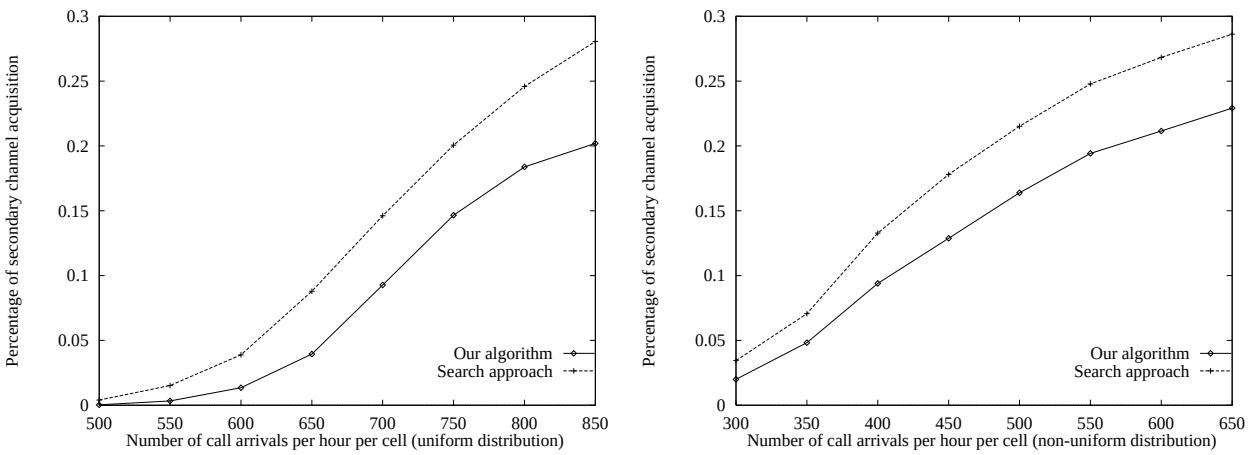

Figure 5: Percentage of secondary channel acquisition

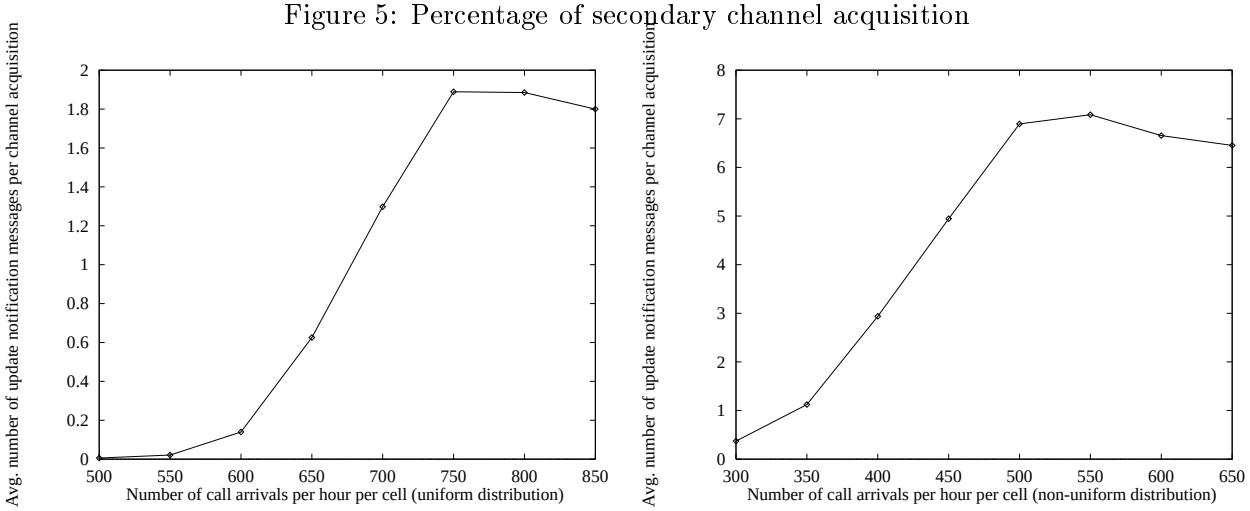

Figure 6: The number of update notification messages

algorithm still has lower message complexity than the search approach most of time. This can be explained by the fact that both algorithms have different secondary channel acquisition percentage. From Figure 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, the search approach does not consider resource planning. Thus, a cell is more likely to run out of primary channels in the search approach than in our algorithm. Also, in our algorithm, a cell only borrows a channel from the "richest" interference neighbors, which reduces the chance that the lender might soon use up its primary channels and have to acquire a secondary channel, but in the search approach, a cell just randomly borrows a channel from its neighbors.

The update notification messages are shown in Figure 6. Note that if we use the Geometric strategy or the update approach as the underlying channel selection algorithm, there is no update notification message overhead, but the failure rate will be the same as the Geometric strategy or

the update approach. Then, the message overhead of our algorithm will be the number of messages shown in Figure 4 subtract the number of update notification messages shown in Figure 6, which is always lower than the search approach even under non-uniform traffic distribution.

The update notification message complexity is determined by the number of secondary channels and other factors, such as how long a borrower keeps the borrowed channel. In general, a borrower keeps the borrowed channel longer under non-uniform traffic than that under uniform traffic. Thus, there are more update notification messages under non-uniform traffic than those under uniform traffic. An interesting fact is that the number of update notification messages drops after some point (750 under uniform traffic). This can be explained by the fact that the failure rate increases faster than the secondary channel acquisition percentage after that point.

From the comparison of message complexity, we observe an interesting result: even though our channel allocation algorithm is fault-tolerant, it still has lower message complexity than known non-fault-tolerant distributed channel allocation algorithms [6, 16].

5.3 Simulation Results (With Failures)

In this section, we compare the performance of the fault-tolerant and non-fault-tolerant channel allocation algorithms under non-uniform traffic distribution.

5.3.1 Failure Rate Under MSS Failures or Network Partitioning

If an MSS fails, every call requests in that cell fails. 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 a MSS fails *does not* necessary mean that it crashes. A failed MSS may still be able to support its MHs, but it cannot communicate with other MSSs, and then it cannot borrow (lend) channels from (to) other MSSs.

Figure 7 compares the failure rate of the fault-tolerant and non-fault-tolerant algorithms under four conditions: no failure, one MSS fails, three MSSs fail, and seven MSSs fail. As shown in Figure 7, the failure rate of the non-fault-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 fault-tolerant channel allocation algorithm, a cell can still borrow channels even if it cannot communicate with some interference neighbors, but in the non-fault-tolerant channel allocation algorithm, a cell cannot borrow any channel from its neighbors when it cannot communicate with any of its interference neighbors. 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. Then, a cell can only use its primary

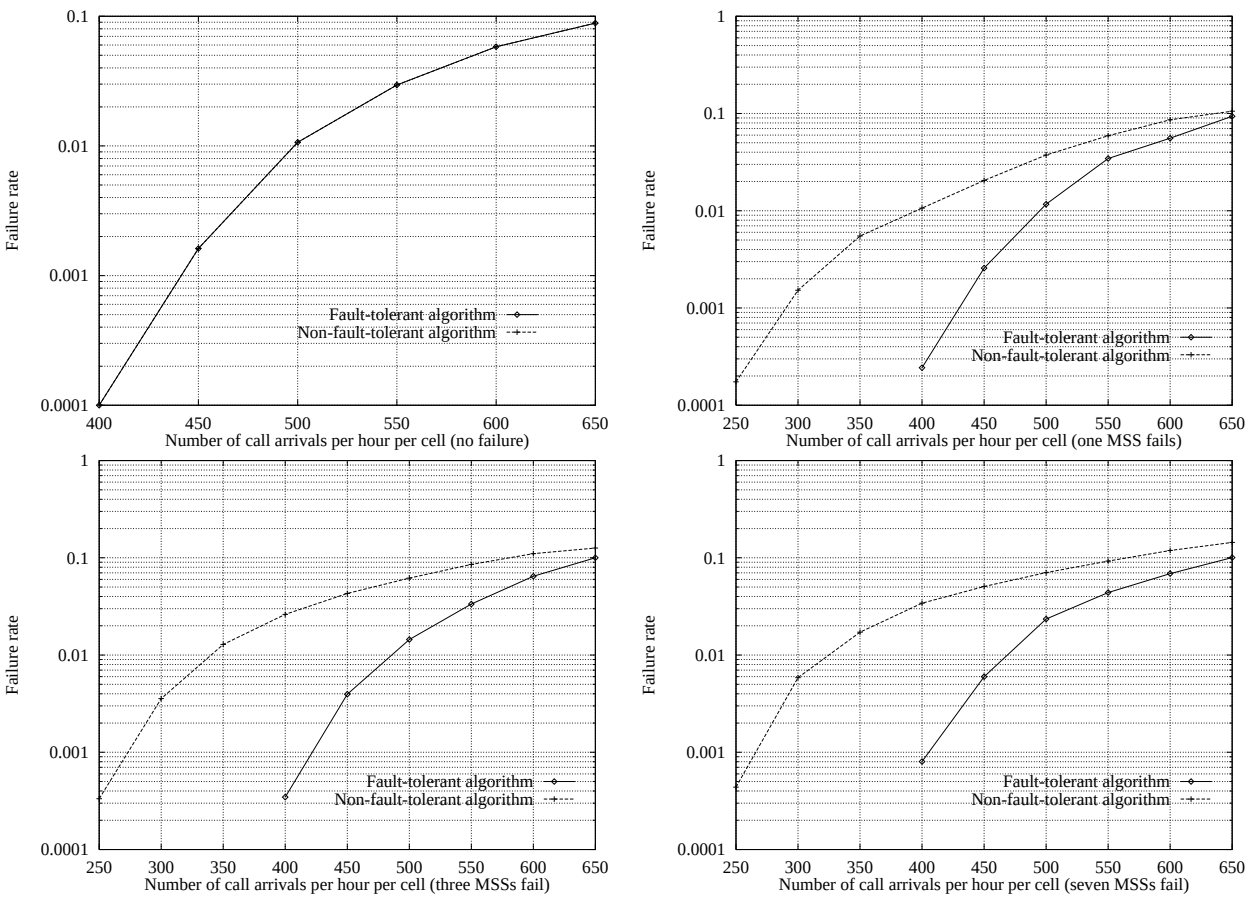

Figure 7: Failure rate under MSS or communication link failures

channels when it cannot communicate with any of its 30 interference neighbors. Thus, the non-fault-tolerant algorithm has a high failure rate under MSS failures. For the fault-tolerant algorithm, 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.

From Figure 7, 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 degrada-*

tion, which is highly desirable in fault-tolerant distributed 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.

5.3.2 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, etc. Since an MSS has other functionalities besides handling channel borrowing, it is very difficult to model it. To simplify the model, we assume network congestion occurs when a node is in hot state. It has an exponentially distributed time with mean 60s. We assume that the communication delay is 4 ms during a network congestion. The maximum tolerable delay of an inter-cell handoff is much less than that of a call request. We assume the maximum tolerable delay of an inter-cell handoff is 10 ms.

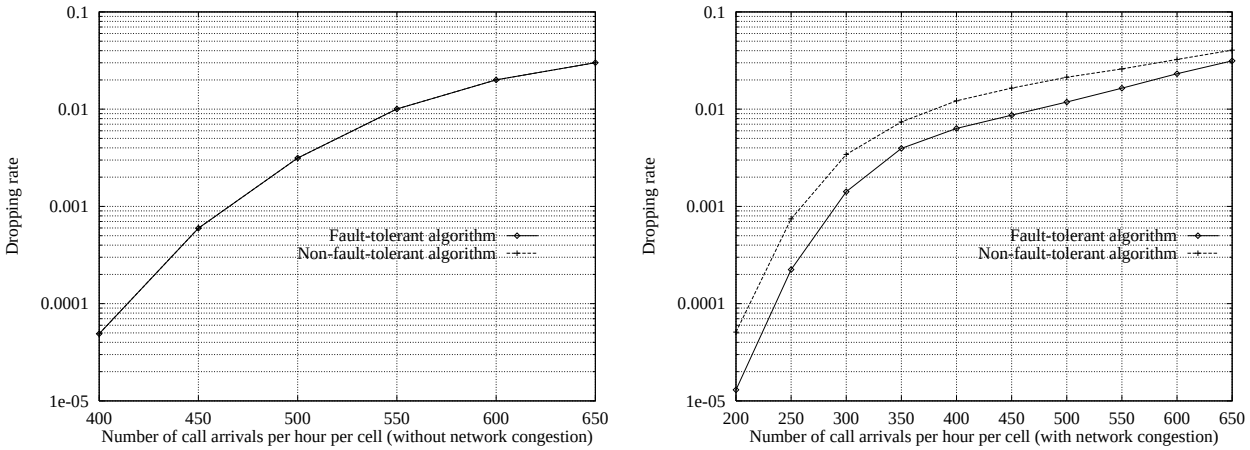

Figure 8: Dropping rate under network congestion

Since the non-fault-tolerant algorithm needs to wait for every interference neighbor, it needs more than $4 * 4 = 16ms$ to borrow a channel from its neighbors, which is longer than the maximum tolerable delay 10 ms. Thus, the handoff requests are dropped during network congestion. However, in our algorithm, a cell can still borrow a channel from other neighbors which does not suffer from network congestion. In this case, it needs only $2 * 4 = 8ms < 10ms$ to borrow a channel. This explains why the fault-tolerant algorithm has lower handoff dropping rate than the non-fault-tolerant algorithm does as shown in Figure 8.

From Figure 8, we can also see that the fault-tolerant channel allocation algorithm under network congestion has very higher dropping rate compared to the dropping rate without network congestion. This can be explained as follows. Network congestion occurs when a cell is in hot state, which is reflected by low inter-handoff rate. Suppose C_i is suffering from network congestion. It's

more likely that most of C_i's neighbors are still in normal state, which has high inter-handoff rate. In other words, there are many more MHs coming from C_i's neighbors than those leaving C_i. Since C_i is experiencing network congestion, any communication with outside cell will have long delay, which results in high dropping rate. Certainly, it is still much lower than the dropping rate in the non-fault-tolerant algorithm. Based on the above reasoning, if a cell can temporally keep the borrowed channel for some time (a system tuning factor), the dropping rate of our algorithm may be reduced. Also, if we use a different model (i.e., network congestion occurs randomly), the dropping rate of the fault-tolerant channel allocation algorithm will be much lower, since the congested cell may not be in hot state. However, the dropping rate of the non-fault-tolerant channel allocation algorithm will not change since the borrower needs to communicate with its congested interference neighbors even though they are in hot states.

6 Conclusions

Distributed channel allocation algorithms have received considerable attention because of their high reliability and scalability. However, previous algorithms cannot tolerate any communication failures and node failures. Moreover, they have poor performance under network congestion, which is very common under heavy traffic load.

In this paper, we proposed a fault-tolerant channel acquisition algorithm which tolerates communication link failures and MSS 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 are carried out in order to evaluate our proposed methodology. Simulation results show that our algorithm significantly reduces failure rate under network congestion, communication link failures, and node crashes compared to non-fault-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 non-uniform 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 Transactions on Vehicular Technology*, pages 14–23, February 1995.

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

[3] S.K. Das, S.K. Sen, and R. Jayaram. "A Dynamic Loadbalancing Strategies for Channel Assignment Using Selective Borrowing in Cellular Mobile Environments". *ACM/Baltzer Wireless Networks (WINET)*, 3:333–347, 1997.

[4] S.B. Davidson, H. Garcia-Molina, and D. Skeen. "Advanced Mobile Phone Service: The Cellular Concept". *Bell System Technical Journal*, pages 15–41, January 1979.

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

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

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

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

[9] H. Jiang and S. Rappaport. "Prioritized Channel Borrowing Without Locking: A Channel Sharing Strategy for Cellular Communications". *IEEE/ACM Transactions on Networking*, 4:163–172, April 1996.

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

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

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

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

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

[15] L. Ortigozo-Guerrero and D. Lara-Rodriguez. "A Compact Pattern With Maximized Channel Borrowing Strategy For Mobile Cellular Networks". *The 7th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, pages 329–333, 1996.

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

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

[18] W. Stalling. "Data and Computer Communications". *Prentice-Hall*, 1996.

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

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