2006

FAULT-TOLERANT DISTRIBUTED CHANNEL ALLOCATION ALGORITHMS FOR CELLULAR NETWORKS

Jianchang Yang
University of Kentucky, yang@fredonia.edu

ABSTRACT OF DISSERTATION

Jianchang Yang

The Graduate School
University of Kentucky
2006

FAULT-TOLERANT DISTRIBUTED CHANNEL ALLOCATION ALGORITHMS FOR CELLULAR NETWORKS

ABSTRACT OF DISSERTATION

A dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy in the
College of Engineering
at the University of Kentucky

By
Jianchang Yang
Lexington, Kentucky
Director: Dr. D. Manivannan, Associate Professor of Computer Science
Lexington, Kentucky
2006

ABSTRACT OF DISSERTATION

FAULT-TOLERANT DISTRIBUTED CHANNEL ALLOCATION ALGORITHMS FOR CELLULAR NETWORKS

In cellular networks, channels should be allocated efficiently to support communication between mobile hosts. In addition, in cellular networks, base stations may fail. Therefore, designing a fault-tolerant channel allocation algorithm is important. That is, the algorithm should tolerate failures of base stations. Many existing algorithms are neither fault-tolerant nor efficient in allocating channels.

We propose channel allocation algorithms which are both fault-tolerant and efficient. In the proposed algorithms, to borrow a channel, a base station (or a cell) does not need to get channel usage information from all its interference neighbors. This makes the algorithms fault-tolerant, i.e., the algorithms can tolerate base station failures, and perform well in the presence of these failures.

Channel pre-allocation has effect on the performance of a channel allocation algorithm. This effect has not been studied quantitatively. We propose an adaptive channel allocation algorithm to study this effect. The algorithm allows a subset of channels to be pre-allocated to cells. Performance evaluation indicates that a channel allocation algorithm benefits from pre-allocating all channels to cells.

Channel selection strategy also influences the performance of a channel allocation algorithm. Given a set of channels to borrow, how a cell chooses a channel to borrow is called the channel selection problem. When choosing a channel to borrow, many algorithms proposed in the literature do not take into account the interference caused by borrowing the channel to the cells which have the channel allocated to them. However, such interference should be considered; reducing such interference helps increase the reuse of the same channel, and hence improving channel utilization. We propose a channel selection algorithm taking such interference into account.

Most channel allocation algorithms proposed in the literature are for traditional cellular networks with static base stations and the neighborhood relationship among the base stations is fixed. Such algorithms are not applicable for cellular networks with mobile base stations. We propose a channel allocation algorithm for cellular networks with mobile base stations. The proposed algorithm is both fault-tolerant and reuses channels efficiently.

KEYWORDS: distributed channel allocation, resource planning, fault-tolerance, cellular networks, 3-cell cluster model.

Jianchang Yang

FAULT-TOLERANT DISTRIBUTED CHANNEL ALLOCATION ALGORITHMS FOR CELLULAR NETWORKS

By
Jianchang Yang

Director of Dissertation
Dr. D. Manivannan

Director of Graduate Studies
Dr. Grzegorz W. Wasilkowski

RULES FOR THE USE OF DISSERTATIONS

Unpublished dissertations submitted for the Doctor's degree and deposited in the University of Kentucky Library are as a rule open for inspection, but are to be used only with due regard to the rights of the authors. Bibliographical references may be noted, but quotations or summaries of parts may be published only with the permission of the author, and with the usual scholarly acknowledgments.

Extensive copying or publication of the dissertation in whole or in part also requires the consent of the Dean of the Graduate School of the University of Kentucky.

A library that borrows this dissertation for use by its patrons is expected to secure the signature of each user.

Name	Date

DISSERTATION

Jianchang Yang

The Graduate School
University of Kentucky
2006

FAULT-TOLERANT DISTRIBUTED CHANNEL ALLOCATION ALGORITHMS FOR CELLULAR NETWORKS

DISSERTATION

A dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy in the
College of Engineering
at the University of Kentucky

By
Jianchang Yang
Lexington, Kentucky
Director: Dr. D. Manivannan, Associate Professor of Computer Science
Lexington, Kentucky
2006

ACKNOWLEDGMENTS

I thank many people. Without their help, this dissertation will not be the way it is.

First, I thank Dr. D. Manivannan, my Ph.D advisor and the chairman of my Ph.D committee. I am extremely lucky to work under his guidance. He gives me countless help and advice. Discussing with him is a real pleasure. His opinion is unique and his comments are inspiring. I really appreciate his support, encouragement, kindness and patience. Without him, I would not be me today and this dissertation would not be possible.

I express my gratitude to other committee members, Dr. Mukesh Singhal, Dr. Jun Zhang, Dr. James Robert Heath, and Dr. Hank Dietz, for their valuable comments and suggestions.

Further, I thank Qiangfeng Jiang for his valuable input. It is my pleasure to work together with him in the same lab. I have learned a lot through discussion with him.

Finally, I thank my wife. She gives me full support.

Table of Contents

List of Tables

List of Figures

List of Files

1. JianchangYang-Dissertation.pdf

Chapter 1

Introduction

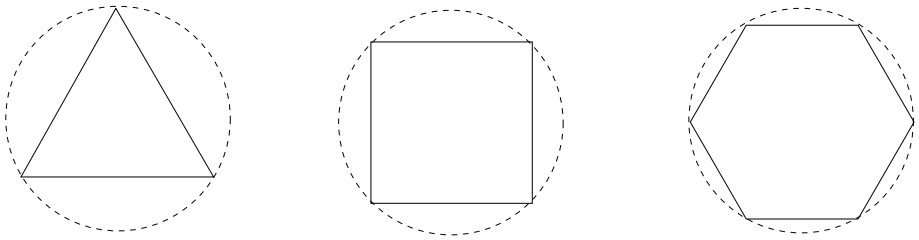
Recent advances in wireless communication technology and portable computing devices have enabled rapid development of mobile computing systems. In mobile computing systems, mobile nodes are equipped with wireless interfaces, and they remain connected to the network through wireless links even when they are mobile. There are several network models of mobile computing systems, including wireless Local Area Networks (wireless LANs), mobile ad-hoc networks (MANET), and cellular networks. In the following, we will briefly explain wireless LANs and MANETs. Then, we will focus on cellular networks and channel allocation algorithms for cellular networks.

Local Area Networks(LANs) are used to interconnect computers in a relatively small area using wires or cables. In wireless LANs, computers communicate by means of high-frequency radio waves. There are several wireless LAN standards, among which, IEEE 802.11 (including 802.11, 802.11a, 802.11b, and 802.11g) is widely used. The IEEE 802.11 standard defines a structure called a basic service set (BSS). There are two types of BSSs [35]. The first type is called an independent BSS (IBSS), which is the most basic type. Stations (or mobile devices) communicate with each other without support of any fixed infrastructure. This is an ad hoc model. We will discuss this type in more detail later. In the second type, a special station, *access point* (AP), exists in each BSS. The access point acts as a relay to transfer messages from source computers to destination computers. Several access points may combine to form an extended network, or may be connected to external wired LANs. This type of wireless network is widely used in practice.

As mentioned earlier, one of the two types that the IEEE 802.11 standard supports is the ad hoc model. This model in wireless LANs is a special case of MANET. In MANET, a collection of mobile devices (or nodes) join together to form a network, without any support from fixed

infrastructures. The link between nodes is wireless. Due to the limitation of transmission range, a node may not be within the transmission range of every other node in the network. In order for the network to be functional, nodes also serve as routers, and they cooperate with each other in forwarding packets from one node to the other. Therefore, a MANET can be viewed as a multi-hop, peer-to-peer mobile wireless network, where packets from source nodes get transfered to destination nodes, via intermediate nodes [1].

In cellular networks, the entire geographical area covered by a cellular network is divided into smaller regions called *cells*. Typically, there is a base station (BS) at the center of each cell. The shape of a cell in a real cellular network is usually irregular. Ideally, it should be a circle, assuming that the base station at the center of the cell has an omni-directional antenna and the transmission power of the antenna in all directions is equal. However, an infinite two-dimensional space cannot be tessellated by circles. Among all geometric shapes, only hexagon, square, and equilateral triangle can be used to tessellate an infinite two-dimensional space [35]. Among these three shapes, hexagons are the most commonly used to represent cells, because they approximate circular regions covered by omni-directional antennas better [35]. These different shapes used to approximate a circular region are shown in Figure 1.1.



(a) an equilateral triangle (b) a square (c) a hexagon

Figure 1.1: Different cell shapes

The base station (BS) at the center of each cell provides communication service to the mobile hosts (MH) in its cell. Base stations are usually connected by wired links, while the communication links between a base station and mobile hosts in the same cell are wireless. This architecture is shown in Figure 1.2. To make it simple, we only show the wireless links between one base station and its mobile hosts. Moreover, the wired links that connect base stations are not shown.

First, let us explain how communication between two mobile hosts take place. For example, in Figure 1.2, suppose that mobile host A in cell i wants to communicate with mobile host B

in cell j. Mobile host A sends a call request message to the base station in cell i. To support this call request, first the base station in cell i needs to know the current location of mobile host B. This is a location management problem. We are not interested in this problem. A survey of location management algorithms can be found in [21, 37]. We are concerned about how to allocate channels to support communication between mobile hosts that are already located.

In the above example, to support communication between mobile host A and mobile host B, wireless channels must be allocated. There are two kinds of wireless channels: *communication channel* and *control channel*. The former is used to support communication between a mobile host and the base station in the same cell, while the latter is set aside to be used exclusively to send control messages that are generated by the channel allocation algorithm. In this dissertation, unless specified otherwise, the term "channel" or "wireless channel" refers to a communication channel. In the example mentioned above, to support communication between mobile host A and mobile host B, two channels need to be allocated: one for communication between mobile host A and its base station, the other for communication between mobile host B and its base station. The term "channel" is an abstraction of the wireless link which carries information of communication. In cellular networks, the wireless link is usually radio link. The radio spectrum allocated to cellular networks is limited and it is usually divided into a set of non-interfering radio channels. Many techniques are available to divide a given radio spectrum into radio channels, among which, three are most widely used: *frequency division multiple access* (FDMA), *time division multiple access* (TDMA), and *code division multiple access* (CDMA).

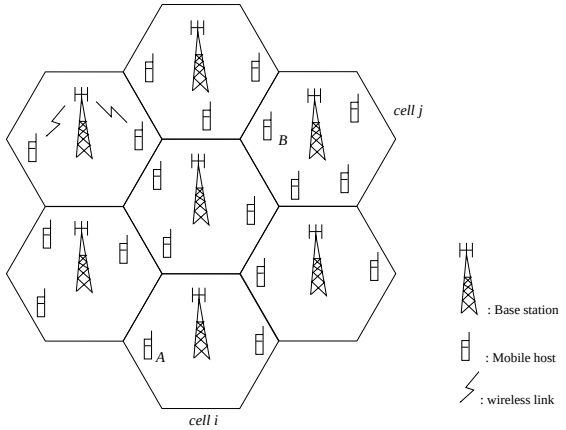


Figure 1.2: An example of a cellular network

FDMA: In FDMA, the radio spectrum is divided into different frequencies, with a guard band between any two adjacent frequencies to eliminate the crosstalk interference. Each user is assigned a frequency for use [35].

TDMA: In TDMA, the usage of the radio spectrum is divided into several time intervals, called *time slots*. User data is transmitted through the time slots assigned to the user [35].

CDMA: In CDMA, channels are differentiated by using different modulation codes. Users share the entire system bandwidth for transmission, and each user is assigned a unique user-signal code [35].

A wireless channel cannot be used in two cells concurrently if the geographical distance between them is less than a threshold distance, called *minimum channel reuse distance (D_{min})* [2, 28], because they will interfere with each other. Such an interference is called *co-channel interference* [32, 33] (it is assumed that wireless channels are orthogonal to one another, so the only interference we need to consider is co-channel interference). A cell, say C_i, is said to be an *interference neighbor* of another cell, say C_j, if the geographical distance between them is less than D_{min}. So if a channel r is used in a cell C_i, then none of C_i's interference neighbors can use r concurrently. Otherwise, co-channel interference arises. If using a channel causes no interference in a cell, then we say that this channel is *available* for the cell.

Wireless channels are a limited system resource, so they should be reused as much as possible. In cellular networks, base stations are responsible for allocating channels to satisfy requests from mobile hosts in their cells. When a mobile host in a cell wants to communicate with another mobile host, it sends a call request to the base station in the cell. Upon receiving such a request, the base station tries to allocate a channel for this mobile host. If a channel can be allocated, then the call request is accepted; otherwise, the call request is dropped. The base stations use a channel allocation algorithm to allocate channels.

A mobile host MH_i which is originally in a cell C_i can move from C_i to one of C_i's neighbors, while involved in a communication with some other mobile host. When such a situation occurs, we say that an *inter-handoff* happens. During an inter-handoff, the channel which is currently supporting the call should be released to the old cell, and a new channel should be allocated in

the new cell. If a new channel is successfully allocated in the new cell for the inter-handoff call, then we say that the inter-handoff call is successful. If a channel cannot be allocated in the new cell, the inter-handoff call is dropped. From the perspective of a mobile host, it is more desirable to drop a new call originating in a cell than to drop an inter-handoff call.

In [16], the authors propose two methods to give priority to inter-handoff calls. These two methods are described below.

- *Channel Reservation:* In each cell, there are a certain number (T) of channels that are set aside exclusively for inter-handoff calls. When an MH involved in a communication moves from cell C_i to cell C_j (C_i and C_j are neighbors), any channel available in cell C_j can be used to support this inter-handoff call. If the number of available channels in cell C_j is lower than T, then no new calls originating in cell C_j will be supported until the number of available channels is higher than T. Thus, inter-handoff calls have higher priority to be supported than the new calls originating in a cell.

- *Inter-handoff Queue:* All the channel requests for inter-handoff calls are queued according to *First Come First Served* (FCFS) principle in each cell. If a cell has available channels and its inter-handoff queue is not empty, then the inter-handoff call at the head of the queue is supported using an available channel. The length of the inter-handoff queue, and the period of time during which an inter-handoff call request can be queued, are upper bounded.

These two methods can be used together with our channel allocation algorithms proposed in this dissertation to deal with inter-handoff calls.

Channel allocation algorithms need to be efficient because channels are a scarce resource in cellular networks. That is, the algorithms need to allocate channels in such a way that less calls are dropped. In addition, channel allocation algorithms need to be fault-tolerant because in cellular networks, base stations may fail, and the links between base stations may break. A fault-tolerant channel allocation algorithm can still perform reasonably well even in the presence of such failures, while a non-fault-tolerant algorithm may cause the system performance degrade badly. Therefore, channel allocation algorithms need to be fault-tolerant in order to improve the availability and performance of the system.

In summary, channels are a limited resource for cellular networks. Therefore they must be allocated efficiently to minimize the dropped calls. In addition, fault-tolerance should also be taken into account when designing channel allocation algorithms. A good channel allocation algorithm for cellular networks should be both efficient and fault-tolerant. In this dissertation, we use the terms "cell" and "base station in the cell" interchangeably.

Next, we discuss the different approaches for channel allocation proposed in the literature.

1.1 Approaches for Channel Allocation in Cellular Networks

Generally, there are two basic approaches for channel allocation:

- **Centralized approach**: In this approach [9, 15, 24, 27, 44, 45, 10, 12], a request for channel allocation is sent to a central controller, called *Mobile Switching Center* (*MSC*). MSC is responsible for allocating channels to cells in such a way that no co-channel interference arises. Since it is a centralized approach, it suffers from the single-point failure problem, i.e., when the MSC fails, the entire system covered by this MSC suffers. In addition, this approach is not scalable because the MSC can become a bottle-neck when the traffic load of the system is very heavy.

- **Distributed approach**: In this approach [39, 38, 42, 5, 6, 7, 40, 43, 17, 41], there is no central controller such as MSC. Instead, base stations share the responsibility to allocate channels. Each base station makes this decision independently, based on its local information, and the information from neighbors. The base station that wants to borrow a channel, and the base station that lends the channel work together to ensure that no co-channel interference arises.

In a distributed channel allocation algorithm, two approaches for exchanging channel usage information, namely, *Search* and *Update*, are usually adopted.

- *Update* [13, 4, 3]: In this approach, a cell notifies its interference neighbors about its current channel usage information whenever its channel usage information changes (i.e., whenever it acquires or releases a channel). So each cell is aware of the set of channels used by its

neighbors. When a cell needs a channel, if some channels allocated to it are available, it just picks one such channel for use. Otherwise, it picks a channel that is not used by any of its interference neighbors and consults with them on whether it can use this channel. If all interference neighbors agree, then it can use the channel. The advantage of this approach is that a cell responds to a call request very quickly since it knows the channel usage information of its interference neighbors. However, this approach suffers from a high message overhead due to the exchange of channel usage information whenever the status of channel utilization changes. A cell notifies its interference neighbors about its change on channel usage even if its neighbors do not need this information at that time. Therefore, some channel usage information may be exchanged unnecessarily. Moreover, the message overhead will be much higher when the system has a very heavy load, when cells acquire and/or release channels frequently.

- *Search* [5, 6, 7, 33, 40, 43]: In this approach, cells exchange channel usage information only when it is necessary. When a cell needs to borrow a channel, it sends a request message to its interference neighbors, asking for their channel usage information. Upon receiving such a request message, a neighbor sends back a reply message which contains its channel usage information. Based on the information the cell receives in the replies, it computes the set of channels that can be borrowed. It picks one such channel and sends messages to its interference neighbors to borrow the channel. If all the neighbors to which the channel has been allocated agree to lend the channel, then the channel borrowing process is complete. This approach does not exchange channel usage information unnecessarily. However, it may respond to a call request slower than the Update approach since first it needs to get the channel usage information of its interference neighbors, and then borrow a channel.

Channel allocation algorithms are usually studied under the following two models:

- *Resource Planning Model* [13, 5, 6, 7, 40]: The set of all cells is partitioned into k disjoint subsets, S_0, S_1, ..., S_{k-1}, in such a way that the geographical distance between any two cells in the same subset is at least D_{min}. The set of all channels available in the system is divided into k disjoint subsets correspondingly, PC_0, PC_1, ..., PC_{k-1}. Channels in PC_i are pre-allocated to cells in S_i and are called *primary channels* of cells in S_i, and *secondary channels*

of cells in S_j ($j \neq i$). When allocating a channel to support a call, a cell, say C_i, always selects a primary channel if one is available. A secondary channel is selected by C_i only when no primary channel is available. If C_i selects a primary channel, it can use this channel without consulting with any neighbor. Otherwise, C_i needs to consult with the neighbors to which the selected secondary channel has been pre-allocated (i.e., the selected secondary channel is a primary channel of these interference neighbors). If all these interference neighbors agree to lend this channel to C_i, then C_i can use this borrowed secondary channel to support the call. After a call using a secondary channel terminates, the secondary channel must be returned to the neighbors to which it has been pre-allocated.

- *Non-Resource Planning Model* [32, 33, 43, 22]: In this model, all the channels are kept in a pool which is known to every cell. Channels are not pre-allocated to any cell. Whenever a cell, say C_i, needs a channel to support a call, it first checks whether there is any channel which is allocated to it and is not being used. It picks one if such a channel exists. Otherwise, it sends a request message to all its neighbors asking for their channel usage information. Based on the information it receives from its neighbors, it begins to compute the set of channels that it can borrow. If the set is not empty, first it tries to pick a channel that is not allocated to any of its neighbors or itself. It no such channel exists, then it tries to borrow a channel that is allocated to some of its neighbors but not being used by any of these neighbors. It consults with its neighbors on whether it can borrow this channel to use. If all the neighbors to which the selected channel has been allocated agree to lend this channel to C_i, then C_i can use this borrowed channel to support the call. After a call using a borrowed channel terminates, the borrowed channel is not returned to any of its neighbors.

In distributed channel allocation algorithms, base stations allocate channels independently. The execution of each base station is characterized by a sequence of events. When base stations exchange messages, sending a message is one event and receiving a message is one event. It is very important to be able to ascertain order between events. One widely used technique to do this is using Lamport's timestamp [23]. Lamport defined the *happened before relation* between events, denoted by $\rightarrow$, to capture causal dependencies between events. The relation $\rightarrow$ is defined as follows [23, 36]:

- $a \rightarrow b$, if a and b are events in the same process and a happened before b.
- $a \rightarrow b$, if a is the event of sending a message m in a process and b is the event of receiving the same message m in another process.
- If $a \rightarrow b$ and $b \rightarrow c$, then $a \rightarrow c$.

Lamport's timestamp can be implemented as follows to order events. Each process P_i has a clock C_i. When an event a occurs at P_i, clock C_i assigns a number $C_i(a)$ to a, called the timestamp of event a at P_i. Two conditions need to be met by system of clocks [36].

[C1] For any events a and b at process P_i, if $a \rightarrow b$, then $C_i(a) < C_i(b)$.

[C2] If a is the event of sending a message m at process P_i and b is the event of receiving the same message m at process P_j, then $C_i(a) < C_j(b)$.

Two implementation rules for the clocks ensure that conditions **C1** and **C2** are met [36].

[R1] C_i is incremented between any two consecutive events in process P_i:
$C_i := C_i + 1$.

[R2] If a is the event of sending a message m by process P_i, then m is assigned a timestamp $t_m := C_i(a)$ (note that the value of $C_i(a)$ is obtained after applying rule **R1**). When a process P_j receives the message m, first it increments C_j by 1, i.e., $C_j := C_j + 1$. Then it updates C_j as follows: $C_j := max(C_j,\ t_m + 1)$.

Combined with unique process id, Lamport's timestamp can be used to totally order all the events. The ordering relation, denoted by $\Rightarrow$, can be defined as follows [36]:

Let a be an event at process P_i and b be an event at process P_j, $a \Rightarrow b$ if and only if

- $C_i(a) < C_j(b)$ or
- $C_i(a) = C_j(b)$ and id of $P_i <$ id of P_j.

In cellular networks, each base station can be viewed as a process. In our dissertation, we use Lamport's timestamp to order events. Each message is assigned a timestamp, using Lamport's timestamp.

1.1.1 Related Works

In [8], the authors address the channel allocation problem using a typical mutual exclusion method. First, the famous dining philosophers problem is described and followed by an efficient solution to it. A two-phase algorithm is proposed for channel allocation based on the solution to the dining philosophers problem. Channels are divided into several disjoint groups. To acquire a channel, a base station needs to hold a read/write lock on a group first. Whenever a base station finishes its channel acquisition process, it releases the lock on that group. Moreover, when a communication session terminates, the channel supporting the session is released.

Channel allocation algorithms proposed in [4, 3] are similar to that proposed in [8]. Channels are divided into disjoint groups. To get a channel, a base station needs to hold a group first. A base station never keeps the acquired channels (i.e., after a communication session terminates, the channel supporting the session is released). The algorithms proposed in [4, 3] use the Update approach. Whenever a base station acquires or releases a channel, it informs its interference neighbors about this. The algorithms proposed in [8, 4, 3] are not fault-tolerant. If any queried base station fails, then the base station which submits the query cannot get a lock on a group and hence cannot acquire a channel, causing call requests to be dropped.

In [22, 20], the channel allocation problem is viewed as a relaxed mutual exclusion problem, and the Non-Resource Planning Model is assumed. Each cell i maintains two sets: R_i and I_i. R_i, the *request set* for cell i, is the set of cells from which i will request permission for using a channel. I_i, the *inform set* for cell i, is the set of cells that cell i will inform of its channel usage information. When a cell i needs to acquire a channel r, it sends a request message to each cell in set R_i. It acquires channel r only if it receives a grant message from each cell in R_i. When cell i releases channel r, it notifies all the cells in set I_i of this. R_i and I_i are constructed and maintained in such a way that no two cells in each other's interference range will use the same channel concurrently, thus guaranteeing no co-channel interference. These algorithms have two disadvantages. First, they are not fault-tolerant. In order for a cell i to acquire a channel, it needs to get permission from each cell in R_i. If any cell in R_i fails, then there is no way for cell i to acquire any channel. Secondly, a cell never keeps a borrowed channel r; it returns channel r after finishing using r. If a new call request originates in a cell just after the cell releases a channel,

then the cell has to repeat the procedure mentioned above again to acquire a channel.

In [13], the Update approach is adopted. The authors assume the Resource Planning Model. When a cell C_i needs a channel, it selects an available channel r. If r is a primary channel, then it marks r as a used channel, and informs all of its interference neighbors about this. If r is a secondary channel, then it sends a request message to each interference neighbor which has r as a primary channel. If all these neighbors agree to lend channel r to C_i, then C_i can use channel r. Otherwise, C_i needs to find another secondary channel to borrow. Whenever a cell acquires or releases a channel, it informs all its interference neighbors about this. Due to this update feature, the algorithm achieves short channel acquisition delay at the expense of higher message overhead. The algorithm is fault-tolerant because the number of C_i's interference neighbors that have r as a primary channel is lower, compared to the total number of C_i's interference neighbors. Even when most of C_i's interference neighbors fail, C_i may still be able to borrow a channel r, as long as its neighbors that have r as a primary channel do not fail and r is not being used by any of these neighbors.

In [47, 46], the authors propose a distributed channel allocation algorithm that is based on a threshold scheme, called D-CAT. The D-CAT scheme makes use of two thresholds: a heavy threshold and a target threshold. The heavy threshold is used to decide whether a cell is heavy (i.e., overloaded), and to trigger the channel allocation algorithm. The target threshold indicates the target number of free channels that a heavy cell wants to acquire. This algorithm determines the number of free channels and the cells from which a heavy cell should import channels to satisfy its channel demand.

In [11], the authors propose a structured channel borrowing scheme for dynamic load balancing in cellular networks. This algorithm solves the tele-traffic hot spot problem in cellular networks. A hot spot is defined as a stack of hexagonal rings of cells and is regarded as complete if all the cells within it are hot. Load balancing is achieved by using a structured channel borrowing scheme in which a hot cell can borrow channels only from adjacent cells in the next outer ring. Thus, unused channels are moved into a hot spot from its peripheral rings.

In [10], the authors propose a dynamic load balancing scheme for the channel assignment problem in cellular mobile environment. A cell is classified either as *hot* or *cold*, based on the

degree of coldness of the cell. The degree of coldness of a cell is defined as the ratio of the number of available channels to the total number of channels allocated to the cell. In this algorithm, unused channels are migrated from underloaded cells to an overloaded cell, i.e., a hot cell borrows channels from cold cells using a channel borrowing algorithm. Users in a cell are also divided into three types: *new*, *departing*, and *others*. Channel demands from these three types of users form different priority classes. Channel assignment is based on the priority class of channel demand.

1.2 Problems Addressed And Solved In This Dissertation

In cellular networks, channels need to be allocated to support communication between mobile hosts efficiently because there are only a limited number of channels available in the system. In addition, a channel allocation algorithm should take into account failure since base stations may fail and/or the link between base stations may break. The algorithm should be able to tolerate such failures and perform well even in the presence of these failures, that is, the algorithm should be fault-tolerant. Many of the channel allocation algorithms in the literature are either not efficient in allocating channels or not fault-tolerant, or both. In this dissertation, we propose channel allocation algorithms which are both fault-tolerant and more efficient compared to some of the existing algorithms.

Under the proposed algorithms, a cell does not need to receive channel usage information from all of its interference neighbors to borrow a channel, which makes the algorithms fault-tolerant. To borrow a channel, it suffices for a cell to receive channel usage information from a subset of its neighbors. The fault-tolerance feature also increases concurrency since neighboring cells can involve in channel borrowing process concurrently without waiting for others to finish their channel borrowing process. In addition to fault-tolerance, under our algorithms, a cell can lend a channel to multiple cells concurrently as long as no two of these cells are neighbors, thus improving channel reuse.

Several factors affect the performance of a channel allocation algorithm. We observe that a compact channel reuse pattern improves channel utilization and pre-allocating channels to cells helps form a compact channel reuse pattern. However, the effect of channel pre-allocation on the performance of a channel allocation algorithm has not been studied quantitatively in the literature.

We study this effect quantitatively with respect to an adaptive channel allocation algorithm which allows a subset of channels to be pre-allocated to cells. Based on our study, we conclude that channel allocation algorithms benefit from pre-allocating all channels to cells.

In addition to channel pre-allocation, how a cell chooses a channel to borrow also influences the performance of the channel allocation algorithm. When choosing a channel to borrow, many of the algorithms proposed in the literature do not take into account the interference caused by borrowing the channel to the cells which have the channel allocated to them. Such algorithms may not allocate channels efficiently. We propose a new channel selection algorithm which takes into account such interference. The selection algorithm chooses a channel to borrow in such a way that it increases the chance of reusing of the same channel in other cells, and hence improving channel utilization.

Although many channel allocation algorithms have been developed for traditional cellular networks where base stations are static, not much work has been done for cellular networks with mobile base stations. The few existing channel allocation algorithms for cellular networks with mobile base stations are neither fault-tolerant nor efficient. Moreover, some of these algorithms may cause co-channel interference. We propose a fault-tolerant channel allocation algorithm for cellular networks with mobile base stations which reuses the allocated channels efficiently.

1.3 Organization of the Dissertation

The rest of the dissertation is organized as follows. In Chapter 2, we present a fault-tolerant channel allocation algorithm for cellular networks. Under this algorithm, to borrow a channel, a cell does not need to get channel usage information from all its interference neighbors, which makes the algorithm fault-tolerant. A channel can be lent to multiple cells as long as no two of them are interference neighbors, which improves channel utilization. In Chapter 3, we extend the algorithm presented in Chapter 2 to suit for cellular networks with mobile base stations. Due to mobility of base stations, the neighborhood information of each base station changes dynamically, which makes channel allocation algorithms assuming a static neighborhood inapplicable in this new situation. To deal with mobility, the algorithm proposed in Chapter 3 computes the neighborhood information for each base station dynamically. When base stations exchange channel usage infor-

mation, they also exchange their current neighborhood information, which assists base stations to compute the set of channels to borrow. Both algorithms presented in Chapter 2 and Chapter 3 assume that channels are not pre-allocated to cells. We study the effect of channel pre-allocation on the performance of channel allocation algorithm and present the results in Chapter 4. In this chapter, we modify the algorithm proposed in Chapter 2 to allow a subset of available channels to be pre-allocated to cells, while the rest are kept in an open pool. Results from performance evaluation indicate that pre-allocating all channels to cells achieves the best performance. Pre-allocating all channels to cells is an example of Resource Planning Model. Based on the results from Chapter 4, we conclude that Resource Planning Model achieves best performance. In Chapter 5, we present a more general algorithm for channel allocation under Resource Planning Model. The proposed algorithm includes a new channel selection algorithm which, when selecting a channel to borrow, takes into account the interference caused by borrowing the channel to the cells which have the channel allocated to them. The channel selection algorithm chooses a channel to borrow in such a way that it increases the chance of getting the same channel reused in other cells, which helps improve channel utilization. Finally, we conclude the dissertation by outlining our future work in Chapter 6.

Chapter 2

A Distributed, Fault-Tolerant Channel Allocation Algorithm for Cellular Networks Under 3-Cell Cluster Model

2.1 Introduction

In cellular networks, the entire geographical area covered by a cellular network is divided into smaller regions called *cell*s [18]. In each cell, there is one *mobile service station*[1] (*MSS*) [33], which serves the *mobile hosts* (*MH*) in the cell. An MH in a cell communicates with the MSS in the cell directly through a wireless link, while MSSs are usually connected by a wired network. Such an architecture [19] is shown in Figure 2.1.

In order for an MH to talk with another MH, channels must be allocated by MSSs to support this communication. For example, in Figure 2.1, suppose that mobile host MH_i in cell E wants to communicate with another mobile host MH_j in cell B. To support this communication, two wireless channels are needed: one for communication between MH_i and the MSS in cell E, the other for communication between MH_j and the MSS in cell B.

When the MSS in a cell runs out of channels, it can borrow a channel from neighboring MSSs. Most algorithms proposed in the literature that use a Search approach [5, 32, 33] require that in order to borrow a channel, an MSS has to receive channel usage information from each of its interference neighbors. This is not fault-tolerant because in cellular networks, MSSs may fail and the links between MSSs may break. In such cases, an MSS may not be able to borrow a channel even when some channels are available to borrow, because it needs to get channel usage information

[1] A mobile service station (MSS) is the same as a base station (BS).

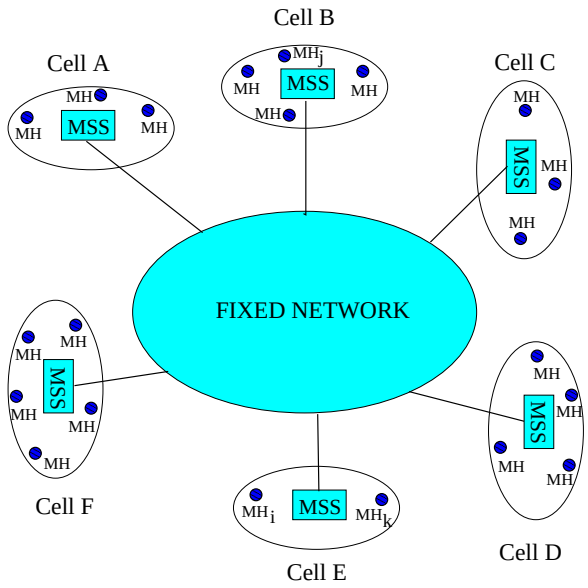


Figure 2.1: A model of a cellular network

from all neighboring MSSs, and such information from some neighboring MSSs may never arrive at it because of link breakage, or some neighboring MSSs may never send their information due to their own failure. Therefore, such a channel allocation algorithm may drop calls unnecessarily.

In this chapter, we propose an efficient, fault-tolerant, distributed channel allocation algorithm under the 3-cell cluster model (described on page 17). Under our algorithm, to borrow a channel, an MSS does not need to receive channel usage information from all neighboring MSSs. It can successfully borrow channels even when some neighboring MSSs have failed. Our algorithm can tolerate mobile service station failures, link failures and messages loss due to network congestion. Moreover, in our approach, a channel can be lent to multiple cells as long as no two of them are interference neighbors. This improves the channel reuse frequency.

The rest of this chapter is organized as follows. In Section 2.2, system model is described. Related works are reviewed in Section 2.3. In Section 2.4, the proposed algorithm is presented in detail, followed by its correctness proof. The performance of the proposed algorithm is evaluated by simulation and compared with existing work in Section 2.5. Section 2.6 summarizes the features of the proposed algorithm.

2.2 System Model

Following are the assumptions made in this chapter:

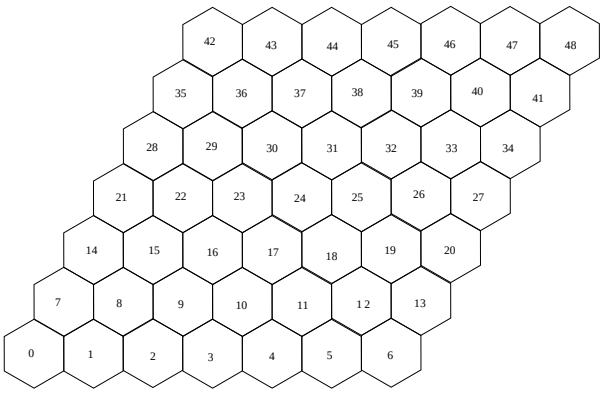


Figure 2.2: A cellular network

- The **3-cell cluster model** is assumed. This model requires that at most one communication session be supported by a channel in a cluster of 3 mutually adjacent cells at any given time. If a channel is being used in a cell, then none of this cell's neighbors can use this channel concurrently. However, the same channel can be used concurrently by cells which are at least two hops away. For example, in Figure 2.2, neighbors of cell 24 are cells 17, 18, 23, 25, 30 and 31. If cell 24 is using a channel r, then none of its six neighbors can use the same channel r concurrently. However, the same channel r can be used concurrently by cell 24 and cell 36, which are two hops away from each other.

- We assume that the cells shown in Figure 2.2 wrap around, thus each cell has 6 neighbors, including the ones at the edge. For example, cell 6 has six neighbors, namely, 5, 12, 13, 0, 42, and 48.

- Both the MSSs and the communication links (the wired or wireless links) could fail. If an MSS in a cell fails, then all the calls supported by it fail at the same time. An MH could fail as well. The failure of an MH only affects its ongoing communication.

All the channels available in the system are kept in an open pool and no channels are pre-allocated to any cell. Therefore, we use Non-Resource Planning Model (described on page 8), in this chapter.

In this chapter, we use the terms "cell" and "the MSS in the cell" interchangeably. When a cell C_i needs to allocate a channel to support a call, C_i first tries to select an unused channel allocated to it to support the call. If C_i has no unused channels allocated to it when a new call request

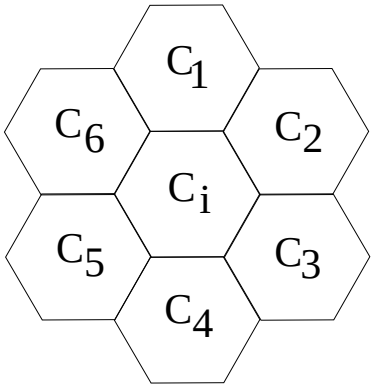


Figure 2.3: Channel borrowing process in cell C_i

originates in the cell, then C_i has to borrow a channel from its neighbors to support the new originating call. To borrow a channel, C_i needs to send request messages to each of its neighbors asking for their channel usage information. In this case, we say that C_i is in *Search Mode* and it is called a *borrower*. When a cell grants a borrower's request for a channel, we call this cell a *lender*. If C_i gets permission to use a channel from all neighbors to which the channel has been allocated, then it can allocate this channel to itself and use this channel to support the call. C_i keeps this channel even after the call using this channel terminates. For example, in Figure 2.3, suppose that cell C_i needs to borrow a channel. It sends request messages to all its neighbors: cells C_1, C_2, C_3, C_4, C_5 and C_6. Suppose that C_i selects a channel r to borrow, where r has been allocated to C_2 and C_5 but not to any other neighbor. If both C_2 and C_5 agree to lend r to C_i, then C_i can use r. C_i keeps the borrowed channel r after the call supported by channel r terminates in C_i. By allowing a cell to keep a borrowed channel, a cell where a lot of calls have originated (i.e., a heavily loaded cell) may have more channels allocated to it. Thus, channels can move from lightly loaded cells to heavily loaded cells, achieving good channel usage.

2.3 Related Works

In [33], the authors propose a distributed dynamic channel allocation algorithm under the 3-cell cluster model. In their approach, channels are not pre-allocated to cells. When a cell needs a channel to support a call, it picks an available channel which is allocated to it to support the call. If no channels allocated to it are free, then it sends a request message to all its neighbors, asking for their channel usage information. Based on the channel usage information received from all its

neighbors, it computes the set of channels that can be borrowed and picks one such channel to borrow. If all the neighbors to which the selected channel has been allocated grant its request, then it allocates the channel to itself and uses this channel to support the call. After a cell grants a neighbor's request for some channel r, it marks this channel for transfer. A marked channel will neither be used by itself nor be lent to grant any other neighbor's request.

Following are some deficiencies of the algorithm proposed in [33].

- *Not fault-tolerant:* To borrow a channel, a cell has to get channel usage information from all its neighbors. This is not fault-tolerant[2] because in cellular networks, a cell may not be able to get this information from all its neighbors due to network congestion and MSS/link failure.

- *Not efficient in channel reuse:* When a cell grants a borrower's request for some channel, it cannot grant a second borrower's request for the same channel even if these two borrowers are not interference neighbors. For example, in Figure 2.4, assume that both cell 7 and cell 3 want to borrow channel r from cell 1. Suppose that cell 1 grants cell 7's request first. Channel r will be marked for transfer, and cell 1 will not grant cell 3's request for the same channel r, since r has already been marked for transfer. However, using r by cell 7 and cell 3 concurrently does not cause co-channel interference because they are not interference neighbors of each other. Thus, the algorithm proposed in [33] does not make efficient reuse of channels.

To our knowledge, Prakash et al.'s algorithm [33] is the only channel allocation algorithm that uses the 3-cell cluster model. So we compare our algorithm with their algorithm. Our proposed channel allocation algorithm has the following advantages over the algorithm proposed in [33].

- *Fault-tolerant:* Our algorithm is fault-tolerant since a cell does not need to receive channel usage information from all its neighbors to borrow a channel. It can borrow a channel from its neighbors as long as it has received channel usage information from all members of some

[2]In Section IX of [33], the authors address the fault-tolerance issue. However, fault-tolerance is achieved based on some assumptions. For example, in [33], if MSS_j has not received $REPLY$ message from a neighbor MSS_i after a timeout period, it assumes that: (i) MSS_i has failed, (ii) MSS_j has received a $REPLY$ from MSS_i such that $Allocate_i := Busy_i := Transfer_i := \emptyset$. By making such an assumption, co-channel interference may arise when both MSS_i and MSS_j select the same channel to use. Thus, the fault-tolerance issue is not really solved in [33].

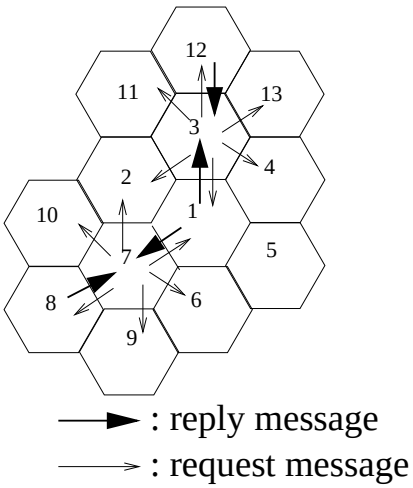


Figure 2.4: Multiple borrowers asking for the same channel

group (a subset of its neighbors), and a channel that has been allocated to all members of the group which is not being used by any of them exists.

- *Efficient in channel reuse:* In our algorithm, a channel can be lent to multiple cells (at most three) as long as no two of them are neighbors, which makes our algorithm achieve better channel reuse.

2.4 A Fault-Tolerant Distributed Channel Allocation Algorithm

In this section, we present our algorithm in detail. First, we explain the basic idea behind our algorithm, then we describe the algorithm formally.

2.4.1 Basic Idea

As mentioned earlier, our algorithm uses the 3-cell cluster model. Each cell C_i has a unique id and has six neighbors; each neighbor is given a unique neighbor id, ordered from 1 to 6, i.e., *nb_1, nb_2, . . . , nb_6*. The set of neighbors of C_i is denoted by NB_i, i.e., $NB_i := \{nb\_1, nb\_2, nb\_3, nb\_4, nb\_5, nb\_6\}$. This is shown in Figure 2.5.

The set of neighbors of C_i are divided into five groups: (note that these groups are not pairwise disjoint):

1. $Group\_1$: $\{nb\_1, nb\_4\}$;

2. $Group\_2$: $\{nb\_2, nb\_5\}$;

3. $Group\_3$: $\{nb\_3, nb\_6\}$;

4. $Group\_4$: $\{nb\_1, nb\_3, nb\_5\}$;

5. $Group\_5$: $\{nb\_2, nb\_4, nb\_6\}$.

Channels are not pre-allocated to any cell. They are uniquely ordered: the channel with the lowest frequency has the minimum order, while the channel with the highest frequency has the maximum order [33].

When a cell C_i needs a channel to support a call, first it checks if there is any channel allocated to it that is not being used. If such a channel exists, it picks the one with the highest order to support the call. Otherwise (i.e., all channels allocated to C_i are being used), it sends a request message to all its neighbors asking for their channel usage information, sets a timer, and waits for reply messages. A reply message from a neighbor includes channel usage information of that neighbor. In this case, we say C_i is in Search Mode and it is a borrower. Each message exchanged between cells is assigned a timestamp using Lamport's timestamp [23] (described on page 8). For example, each request message has a timestamp. The lower the timestamp, the higher the priority. Messages can be ordered by their timestamp; ties can be broken by using cell id.

After the timer expires or cell C_i receives replies from all neighbors, C_i begins to compute the set of channels that it can borrow. If it receives replies from all neighbors, then it first computes the set of channels that are not allocated to any of its neighbors or itself. If such a channel exists, it allocates and uses the channel immediately. Otherwise, it computes the set of channels which

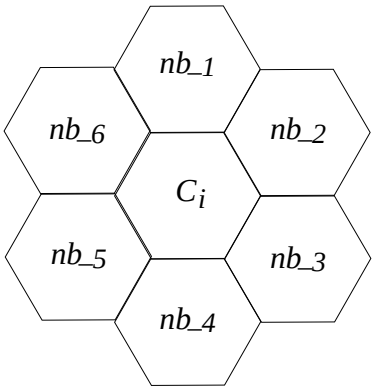


Figure 2.5: Neighborhood relationship in the cellular network

are not being used by any neighbor or itself. If this set is empty, then it drops the call. Otherwise, it selects a channel from the set and consults with the neighbors to which the channel has been allocated to borrow the channel.

If it has not received replies from all its neighbors when the timer expires, it may still be able to borrow a channel from neighbors as long as it receives reply messages from all members of any one group (i.e., $Group\_1$ through $Group\_5$) and there is a channel which is allocated to all members of that group, but is not being used by any of them. The way to compute the set of channels that it can borrow depends on the number of replies that it gets. Next, we present the details regarding how a cell computes the available channels based on the number of replies received before the timer expires. Depending on the number of replies received, a cell computes the available channels as follows.

- *One reply:* It can do nothing but drop the call because it has too little information to borrow a channel.

- *Two replies:* Let x and y be the two neighbors from which it gets replies. Two sub-cases arise.

  - *sub-case 1*: x and y are in the same $Group\_m$ for some $m \in \{1, 2, 3\}$. It computes the set of channels that have been allocated to x and y, but not being used by either of them.

  - *sub-case 2*: x and y are not in the same $Group\_m$ for any $m \in \{1, 2, 3\}$. In this case, there is not enough information to borrow a channel and hence the call has to be dropped.

- *Three replies:* Let x, y and z be the three neighbors from which it receives replies. Three sub-cases arise.

  - *sub-case 1:* x, y and z are in the same $Group\_m$ for some $m \in \{4, 5\}$. It computes the set of channels allocated to all of them that are not being used by any of them.

  - *sub-case 2:* Two of the three neighbors, x, y and z, are in the same $Group\_m$ for some $m \in \{1, 2, 3\}$. This case is handled in the same way as in *sub-case 1* of the case: *Two*

replies.

- *sub-case 3:* x, y and z are not in the same $Group\_m$ for any $m \in \{4, 5\}$ and also no two of the three form a $Group\_m$ for any $m \in \{1, 2, 3\}$. In this sub-case, it drops the call.

- *Four replies:* Let w, x, y, and z be the four neighbors from which it receives replies. Two sub-cases arise.
  - *sub-case 1:* Two of the four neighbors are in the same $Group\_m$ for some $m \in \{1, 2, 3\}$. This case is handled in the same way as in *sub-case 1* of the case: *Two replies.*
  - *sub-case 2:* Three of the four neighbors are in the same $Group\_m$ for some $m \in \{4, 5\}$. This case is handled in the same way as in *sub-case 1* of the case: *Three replies.*

- *Five replies:* This is similar to the case: *Four replies.* It checks if any two of the five neighbors from which a reply message has been received are in the same $Group\_m$ for some $m \in \{1, 2, 3\}$. In addition, it checks whether any three of the five neighbors are in the same $Group\_m$ for some $m \in \{4, 5\}$. If either two or three of the five neighbors form a group with two or three members respectively, then it can compute the set of channels it can borrow.

If the set of channels that it can borrow is empty, it drops the call. Otherwise, it selects a channel from the set to borrow. Next, we illustrate with an example how a cell that has received channel usage information (replies) from only a subset of its neighbors selects a channel to borrow. In Figure 2.5, suppose that cell C_i needs to borrow a channel and it gets replies only from nb_1 and nb_4 (note that nb_1 and nb_4 form $Group\_1$). If there exists a channel r which has been allocated to both nb_1 and nb_4 that is not being used by either of them, then C_i can borrow channel r. Since r has been allocated to nb_1, it could not have been allocated to nb_2 or nb_6, because they are neighbors of nb_1. Similarly, r could not have been allocated to nb_3 or nb_5 because they are neighbors of nb_4. Thus, C_i can allocate r to itself and use r safely if both nb_1 and nb_4 agree to lend channel r.

Suppose that C_i gets replies only from nb_1, nb_3 and nb_5, which form $Group\_4$. If there is a channel r which has been allocated to nb_1, nb_3 and nb_5, but is not being used by any of them, then C_i can borrow channel r. Since channel r has been allocated to nb_1, it could not have been allocated to nb_2 or nb_6. Similarly, nb_4 could not have channel r allocated to itself since r has

been allocated to nb_3 and nb_5. Thus, if nb_1, nb_3 and nb_5 all agree to lend r to C_i, then C_i can allocate r to itself and use r without causing co-channel interference. The case in which cell C_i receives four or five replies is similar.

Thus, even if a cell C_i does not receive replies from all its neighbors, it is possible for cell C_i to borrow a channel. It can borrow a channel from neighbors as long as it receives reply messages from all members of any group and there is at least one channel allocated to all members of the group that is not being used by any member of that group.

A cell is allowed to grant requests from several borrowers for the same channel concurrently, as long as no two of the borrowers are neighbors. This increases channel reuse. A cell does not grant the same channel to two borrowers concurrently if the two borrowers are neighbors. If there are two neighboring cells which are in Search Mode concurrently (i.e., two neighboring borrowers) and each of them borrows a channel, then they are not allowed to borrow the same channel (this is proved in Section 2.4.5).

To summarize, when a cell C_j receives another cell C_i's request message, it sends a reply message to C_i including its channel usage information, if it is not in Search Mode or its request message has lower priority than that of C_i's (Lamport timestamps are assigned to messages, the higher the timestamp, the lower the priority). After C_i receives a reply message from all neighbors or when the timer expires, it begins to compute the set of available channels and picks one (if there is any) to borrow. If it gets replies from all neighbors and the selected channel has not been allocated to any of its neighbors or itself, then it can use this channel immediately and add this channel to the set of channels allocated to it. Otherwise, it has to consult with the neighbors to which the channel has been allocated before it uses this channel since a neighbor may allocate this channel to support a call just after sending a reply message. If all such neighbors grant C_i's request, then C_i can allocate the selected channel to itself and use it. If C_i cannot borrow this channel, it picks another available channel if possible and repeats the procedure mentioned above to borrow a channel. If a channel cannot be borrowed, it drops the call. After a call using a borrowed channel terminates, the borrowed channel is not returned to the lender.

Next, before we present the algorithm, we introduce the data structures used in the algorithm. As mentioned earlier, we use the terms "cell C_i" and "the mobile service station MSS_i in cell C_i"

interchangeably.

2.4.2 Data Structures

The data structures maintained by each cell C_i and their content are given in Table 2.1.

Table 2.1: Data Structures

$Spectrum$:	the set of all channels in the system.
NB_i:	the set of neighbors of cell C_i.
$Allocate_i$:	the set of channels currently allocated to C_i.
$Busy_i$:	the set of channels currently being used by C_i.
$Transfer_i$:	the set of channels marked for transfer by C_i.
$Granted_i$:	a set of sets maintained by C_i.
$Lent_i$:	a set of sets maintained by C_i.

Initially, $Allocate_i$, $Busy_i$, $Transfer_i$, $Granted_i(r)$ and $Lent_i(r)$ are all empty. At any given time, $(Transfer_i \cap Busy_i)$ is an empty set and $Transfer_i \subseteq Allocate_i$. $Granted_i$ is a set of sets maintained by cell C_i. $\forall\ r \in Allocate_i$, $Granted_i(r)$ denotes the set of cells to which C_i has sent an *agree(r)* message. An *agree(r)* message from C_i to C_j means that C_i grants C_j's request to borrow channel r. C_i lends channel r to C_j only when it is notified by C_j to release the channel. $Lent_i$ is a set of sets maintained by C_i. $\forall\ r \in Allocate_i$, $Lent_i(r)$ denotes the set of cells to which C_i has lent r. Each message is assigned a timestamp, using Lamport's timestamp [23] (described on page 8). At each cell C_i, obsolete messages from a cell C_j can be detected by comparing their timestamps with the largest timestamp from C_j seen so far by C_i.

In the following, we present the proposed channel allocation algorithm formally.

2.4.3 The Algorithm

A: When a cell C_i needs a channel to support a call,

let $Free_i := Allocate_i - Busy_i - Transfer_i$.

- If $Free_i = \emptyset$, it sends a *request* message to each of its six neighbors and sets a timer.
- If $Free_i \neq \emptyset$, it picks a channel $r \in Free_i$ with the highest order to support the call. $Busy_i := Busy_i \cup \{r\}$. When the call terminates, $Busy_i := Busy_i - \{r\}$.

B: When a cell C_j receives a *request* message from cell C_i,

– If it does not have an outstanding *request* message, or its *request* message has a lower priority than C_i's *request*, then it sends to C_i a *reply* message, including $Allocate_j$ and $Busy_j$ in the *reply* message.

– If its own *request* message has a higher priority than C_i's *request*, it defers sending *reply* to C_i.

C: After a cell C_i receives *reply* messages from all its neighbors or when its timer expires, it sets $Free_i$ to empty. Its subsequent actions depend on the number of *reply* messages (i.e., the number of neighbors responded) that it has received.

– **1: If the number of *reply* messages is equal to six:**

* α: If $Free_i := Spectrum - \cup Allocate_k \neq \emptyset$,
  where $k \in \{nb\_1, nb\_2, nb\_3, nb\_4, nb\_5, nb\_6, i\}$, then a channel
  $r \in Free_i$ with the highest order is selected to support the call.
  $Allocate_i := Allocate_i \cup \{r\}$; $Busy_i := Busy_i \cup \{r\}$;
  $Granted_i(r) := \emptyset$; and $Lent_i(r) := \emptyset$.

* β: Otherwise, $Free_i := Spectrum - \cup Busy_k$,
  where $k \in \{nb\_1, nb\_2, nb\_3, nb\_4, nb\_5, nb\_6, i\}$. If $Free_i = \emptyset$, then it drops the call. Otherwise, it picks a channel $r \in Free_i$ with the lowest order for transfer. It sends a *transfer(r)* message to all those neighbors to which r has been allocated, and sets a new timer.

– **2: If the number of *reply* messages is less than six:**

* α: If the number of *reply* messages is less than two, $Free_i := \emptyset$.

* β: If the number of *reply* messages is equal to two, let x and y ($x \neq y$) be the two neighbors from which a *reply* has been received.

  · If $\exists m \in \{1, 2, 3\}$ such that $\{x, y\} = Group\_m$, then
    $Free_i := \cap_{k \in Group\_m}(Allocate_k - Busy_k)$.

* γ: If the number of *reply* messages is equal to three, let x, y and z be the three neighbors from which a *reply* message has been received.

· If $\exists m \in \{1,2,3\}$ such that $\{x,y,z\} \supset Group\_m$, then

$Free_i := \cap_{k \in Group\_m}(Allocate_k - Busy_k)$.

· Otherwise, if $\exists\ m \in \{4,5\}$ such that $\{x,y,z\} = Group\_m$, then

$Free_i := \cap_{k \in Group\_m}(Allocate_k - Busy_k)$.

* δ: If the number of *reply* messages is equal to four, let x and y ($x \neq y$) be the two neighbors from which a *reply* message has **NOT** been received.

· If $\exists\ m \in \{1,2,3\}$ such that $(NB_i\ -\ \{x,y\}) \supset Group\_m$, then

$Free_{i\_m} := \cap_{k \in Group\_m}(Allocate_k - Busy_k)$.

· If $\exists\ n \in \{4,5\}$ such that $(NB_i\ -\ \{x,y\}) \supset Group\_n$, then

$Free_{i\_n} := \cap_{k \in Group\_n}(Allocate_k - Busy_k)$.

· $Free_i := Free_{i\_m} \cup Free_{i\_n}$.

* θ: If the number of *reply* messages is equal to five, let x be the neighbor from which a *reply* message has **NOT** been received by C_i.

· If $\exists\ m \in \{1,2,3\}$ such that $(NB_i\ -\ \{x\}) \supset Group\_m$, then

$Free_{i\_m} := \cap_{k \in Group\_m}(Allocate_k - Busy_k)$.

· If $\exists\ n \in \{4,5\}$ such that $(NB_i\ -\ \{x\}) \supset Group\_n$, then

$Free_{i\_n} := \cap_{k \in Group\_n}(Allocate_k - Busy_k)$.

· $Free_i := Free_{i\_m} \cup Free_{i\_n}$.

If $Free_i \neq \emptyset$, then it picks a channel $r \in Free_i$ with the lowest order for transfer and sends a *transfer(r)* message to all those neighbors to which r has been allocated and sets a new timer. Otherwise, it drops the call.

D: When a cell C_j receives a *transfer(r)* message from cell C_i,

– If $r \in Busy_j$, it sends a *refuse(r)* message to C_i.

– If $r \in Allocate_j - Busy_j$:

* If $\forall\ C_m \in Granted_j(r)$, C_i and C_m are not neighbors, then it sends an *agree(r)* message to C_i and adds C_i to $Granted_j(r)$. If r is not in $Transfer_j$, then it adds r to $Transfer_j$.

* If $\exists C_m \in Granted_j(r)$ which is a neighbor of C_i , it sends a *refuse(r)* message to C_i.

– If $r \notin Allocate_j$ and $\forall\ C_m \in Granted_j(r) \cup Lent_j(r)$, C_i is not a neighbor of C_m, then it sends an *agree(r)* to C_i and adds C_i to $Granted_j(r)$. Otherwise, it sends a *refuse(r)* to C_i.

E: If C_i receives a response corresponding to each of its *transfer(r)* message before the new timer set in step C expires, it cancels the new timer and does the following:

– If all of them are *agree(r)* messages, then it uses r to support the call. $Allocate_i := Allocate_i \cup \{r\}$; $Busy_i := Busy_i \cup \{r\}$. $Granted_i(r) := \emptyset$; and $Lent_i(r) := \emptyset$. It sends a *release(r)* message to the neighbors from which an *agree(r)* message was received.

– If not all of them are $agree(r)$ messages, it sends a *keep(r)* message to neighbors from which it receives an *agree(r)*. $Free_i := Free_i - \{r\}$, and it tries to borrow another channel if $Free_i \neq \emptyset$.

F: When the new timer set in step C expires before C_i gets a response corresponding to each of its *transfer(r)* message, it drops the call and sends a *keep(r)* message to each neighbor from which it has received an *agree(r)* message.

G: When a cell C_j receives a *release(r)* message from cell C_i,

– $Granted_j(r) := Granted_j(r) - \{C_i\}$; $Lent_j(r) := Lent_j(r) \cup \{C_i\}$.

– If $r \in Allocate_j$, then
$Allocate_j := Allocate_j - \{r\}$; $Transfer_j := Transfer_j - \{r\}$.

H: When a cell C_j receives a *keep(r)* message from cell C_i,

– $Granted_j(r) := Granted_j(r) - \{C_i\}$;

– If $r \in Allocate_j$ and $Granted_j(r) = \emptyset$, then
$Transfer_j := Transfer_j - \{r\}$.

I: After a cell C_i acquires a channel to support a call or it drops a call, it sends a *reply* message to all those request messages for which a reply has been deferred.

2.4.4 An Explanation of the Algorithm

In this section, we briefly explain the working of the algorithm.

When a cell C_i needs a channel to support a call, it first checks whether there is a channel allocated to it that is not being used. If there exists such a channel r, it uses r to support the call, and adds r to the set $Busy_i$. Otherwise (i.e., C_i runs out of channels), it sends a *request* message to each of its six neighbors and sets a timer. It begins to compute the set of available channels that it can borrow either after it receives a *reply* message from all neighbors or after the timer expires.

If it receives *reply* messages from all its neighbors, it first computes the set of channels that are not allocated to any of its six neighbors and itself. If this set is not empty, it picks a channel r from this set to support the call and adds r to both the sets $Allocate_i$ and $Busy_i$. If this set is empty, it computes the set of channels that are allocated to its neighbors but are not being used by any of its neighbors and itself. If such a channel r is found, then it sends a *transfer(r)* message to those neighbors to which r has been allocated. If it gets permission to use r from all these neighbors, then it uses r, adds r to both $Allocate_i$ and $Busy_i$, and notifies these neighbors to release r by sending them a *release(r)* message. Otherwise, it sends a *keep(r)* message to each neighbor from which an *agree(r)* message has been received, and then picks another channel in the set of available channels computed to borrow, if possible, and repeats this procedure.

If the number of *reply* messages that it has received is less than six when the timer expires, its action depends on the number of *reply* messages it gets. If it can compute the set of channels that it can borrow and the set is not empty, then it selects a channel r from this set and sends a *transfer(r)* message to all those neighbors to which r has been allocated. Otherwise, it drops the call.

When a cell C_i successfully borrows a channel r, it sets $Granted_i(r)$ and $Lent_i(r)$ to empty. The set $Granted_i(r)$ keeps track of the set of cells to which C_i has sent an *agree(r)* message, while $Lent_i(r)$ maintains the record of the set of cells from which a *release(r)* message has been

received by C_i. These data structures help in lending a channel r concurrently to multiple cells. We illustrate this with an example using Figure 2.6.

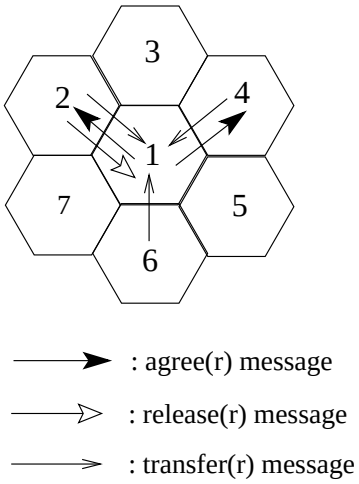


Figure 2.6: An example illustrating multiple borrowing

Suppose $r \in Allocate_1$, $Granted_1(r) := \emptyset$, and $Lent_1(r) := \emptyset$, and cells 2 and 4 send a *transfer(r)* message to cell 1 concurrently. Cell 1 will grant both of their requests according to Step D of the algorithm. Cells 2 and 4 are added to $Granted_1(r)$ and channel r is added to $Transfer_1$. If cell 1 receives a *release(r)* message from cell 2 first, then it removes cell 2 from $Granted_1(r)$ and adds cell 2 to $Lent_1(r)$. In addition, it removes r from $Allocate_1$ and $Transfer_1$ as in Step G of the algorithm. Now the set $Granted_1(r)$ contains cell 4, while the set $Lent_1(r)$ contains cell 2. When cell 1 receives a *release(r)* message from cell 4, it does the similar thing. When cell 2 borrows r, it sets the two sets: $Granted_2(r)$ and $Lent_2(r)$ to be empty. Suppose cell 1 receives a *transfer(r)* message from cell 6 just before receiving the *release(r)* message from cell 4. Then, cell 1 grants cell 6's request since it is not using r and cell 6 is not a neighbor of either cell 2 or cell 4. This helps increase channel reuse because the same channel is lent to three neighbors concurrently.

In our algorithm, when a cell needs a channel and some channels are available, it always selects the channel with the highest order to use. When a cell selects a channel to borrow, it always selects the channel with the lowest order. This channel selection strategy will help reduce the number of *refuse(r)* messages (where r is the selected channel to borrow). For example, in Figure 2.6, suppose that cell 1 has 2 channels available: ch_2 and ch_5, and cell 2 needs to borrow a channel. If the set of channels that cell 2 can borrow consists of ch_2 and ch_5, then cell 2 will select ch_2 to

borrow according to the channel selection strategy of our algorithm. When cell 1 needs a channel, it will select ch_5 first since ch_5 has the highest order among the available channels. Thus, when cell 1 receives cell 2's transfer message for channel ch_2, it will grant cell 2's request to borrow ch_2. Hence, this strategy helps reducing the probability of using a channel r that its neighbors try to borrow.

2.4.5 Correctness of the Algorithm

In this subsection, we prove that the proposed algorithm is correct. First, we prove that two neighboring cells are not allowed to borrow the same channel concurrently under our algorithm. Then, we prove that two neighboring cells are never allowed to use the same channel concurrently. Finally, we prove that our algorithm is deadlock-free[3].

Lemma 2.4.1 *Two neighboring cells, C_i and C_j, are not allowed to borrow the same channel concurrently under our algorithm.*

Proof: Let r_1 be the channel borrowed by C_i , r_2 be the channel borrowed by C_j. Without loss of generality, we assume that C_i's request has lower priority than C_j's request. We prove that $r_1 \neq r_2$. Since C_j's request has higher priority, C_j will defer sending a reply to C_i until it finishes its channel borrowing process. After C_j borrows r_2, it adds r_2 to both $Allocate_j$ and $Busy_j$. Then, it sends a reply message to C_i. The following two cases arise for C_i.

- case 1: C_i borrows r_1 as a result of getting reply messages from all of its neighbors for its request. If $r_1 \in (Free_i := Spectrum - \cup Allocate_k)$ (step C-1-α of the algorithm), then $r_1 \neq r_2$, because $r_2 \in Allocate_j$ and C_j is a neighbor of C_i. If $r_1 \in (Free_i := Spectrum - \cup Busy_k)$ (step C-1-β of the algorithm), then $r_1 \neq r_2$, because $r_2 \in Busy_j$ and C_j is a neighbor of C_i.

- case 2: C_i borrows r_1 as a result of getting reply messages from only a subset of its neighbors for its request. In this case, C_i must have borrowed r_1 from some $Group\_m$ ($m \in \{1, 2, 3, 4, 5\}$). $r_1 \in (Free_i := \cap_{k \in Group\_m}(Allocate_k - Busy_k))$. We have $r_1 \notin Busy_k$ where $k \in Group\_m$. The following two sub-cases arise.

[3]A deadlock occurs when a set of processes in the system is blocked waiting on requirements that can never be satisfied [36].

- sub-case 1: $C_j \in Group\_m$. Because C_j sends a reply message to C_i only after it finishes its channel borrowing process, $r_2 \in Busy_j$. Therefore, $r_1 \neq r_2$ because $r_1 \notin Busy_j$.
- sub-case 2: $C_j \notin Group\_m$. Because C_i and C_j are neighbors, they have two common neighbors. Let these two common neighbors be C_m and C_n. We have $r_1 \in (Allocate_m \cup Allocate_n)$. Assuming that $r_1 = r_2 = r$. There are three sub-cases:
  - (A): $r \in (Allocate_m \cap Allocate_n)$. In this case, C_m and C_n grant both C_i's and C_j's request for channel r. However, this is impossible. According to our algorithm (step D of the algorithm), a cell does not grant requests from two cells for the same channel if they are neighbors. Therefore, $r_1 \neq r_2$.
  - (B): $r \in (Allocate_m - Allocate_n)$. It follows that C_m grants both C_i's and C_j's request for channel r. This is impossible due to the same reason mention above.
  - (C): $r \in (Allocate_n - Allocate_m)$. This is similar to sub-case 2 (B) above.

Thus, two neighboring cells are not allowed to borrow the same channel under our algorithm. □

Lemma 2.4.2 *Two neighboring cells are not allowed to use the same channel concurrently under our algorithm.*

Proof: Initially, no channel is allocated to any cell. Since neighboring cells do not borrow the same channel concurrently, it follows that neighboring cells do not use the same channel concurrently. □

Lemma 2.4.3 *The proposed channel allocation algorithm is deadlock-free.*

Proof: In the proposed algorithm, a timeout mechanism is used. When a cell sends a *request* message or a *transfer(r)* message where r is the selected channel to borrow, it sets a timer. A cell begins to proceed either after it receives response corresponding to each of its messages or after the timer expires. So, hold and wait situation does not arise. Therefore, the algorithm is deadlock-free. □

2.5 Performance Evaluation

2.5.1 Definitions

In this section, we evaluate the performance of our algorithm and also compare it with a modified version of the algorithm proposed in [33]. In [33], in order to borrow a channel, a cell needs to receive channel usage information from each neighbor. This can cause delay when the traffic load is heavy. However, a call request from an MH should be responded to within a reasonable period of time. Thus, we modified the algorithm in [33] to make it respond to a call request in a timely manner. The modifications made to the algorithm in [33] are as follows. When a cell C_i needs to borrow a channel from its neighbors, it sends a *request* message to each neighbor and sets a timer. If the timer times out before it receives channel usage information included in the reply message from all neighbors, then it drops the call. Otherwise (i.e., it receives replies from all its neighbors before the timer expires), it computes the set of channels that can be borrowed. If it finds a channel r that can be borrowed from its neighbors, then it sends a *transfer(r)* message to those neighbors to which r has been allocated, and sets a new timer. If the new timer expires before it can get a response from all those neighbors to which it has sent *transfer(r)* message, then it drops the call. Otherwise, it checks if all the responses are agree(r) messages. If they are, then it can borrow the channel r. Otherwise, it drops the call. The modifications mentioned above allow a cell to respond to an MH's request in a timely manner, and also helps in making a fair comparison with our algorithm.

Four metrics are used to compare the performance of the two algorithms: *call blocking rate*, *handoff drop rate*, *call failure rate* and *message complexity*. Call blocking rate is defined as the ratio of the number of new calls which cannot be supported (i.e., blocked new calls) to the total number of new calls. Handoff drop rate is defined as the ratio of the number of inter-handoff calls dropped to the total number of inter-handoff calls. Call failure rate is defined as the ratio of total number of calls dropped (including blocked new calls and dropped inter-handoff calls) to the total number of calls processed. Message complexity is defined as the ratio of the number of messages exchanged between MSSs and the number of calls processed. This simulation studies the trend that the four metrics change with the increase of call arrival rate, which is defined as the number of call arrivals per hour per cell.

2.5.2 Simulation Parameters

To evaluate the performance of the algorithm, we used CSIM18 [4] Simulation Engine, which is a process-oriented, discrete-event simulator. The simulated cellular network consists of $6 * 6$ cells. Each cell has 6 neighbors (by wrapping around the cells). There are 300 channels total. Initially, no channels are pre-allocated to any cell. We assume that the average one-way communication delay between two cells is 4 milliseconds. This average delay includes transmission delay, propagation delay and the message processing time. The values of the timers used in the algorithm are constants: 8 milliseconds, which is twice as large as the average one-way communication delay. In the simulation, once an MH is generated, it sends a call request to the MSS in the cell. Upon receiving such a request, the MSS tries to allocate a channel to support the call by using the underlying channel allocation algorithm. If no channel can be allocated, then the call is dropped and it is counted as a call failure. If a channel can be allocated to support the call, then the MH will use this channel for its communication. We assume that the average service time per call is 3 minutes. During communication, the MH may move to an adjacent cell (i.e., an inter-handoff occurs). If this happens, it releases the channel which is currently supporting the call to the cell from which it is leaving, and it sends a call request to the cell to which it is moving. The new cell which it is moving into is responsible to allocate a new channel to support the inter-handoff call. If no channel can be allocated for this inter-handoff call, then the call is dropped and counted as an inter-handoff failure. At the end of the simulation, the number of inter-handoff failures, the number of call failures, the number of calls processed, and the number of messages exchanged for channel borrowing are collected.

The simulation is conducted under a non-uniform traffic pattern, which is more realistic. Under such a pattern, a cell can be in one of the two states: normal state and hot state. When a cell is in normal state, call arrival rate is low, and inter-handoff rate is high. When a cell is in hot state, call arrival rate is high and inter-handoff rate is low. The parameters we used for simulating the non-uniform traffic pattern are given in Table 2.2 [6].

[4] CSIM18 Simulation Engine is a product of Mesquite Software, Inc.

Table 2.2: Parameters for non-uniform traffic pattern

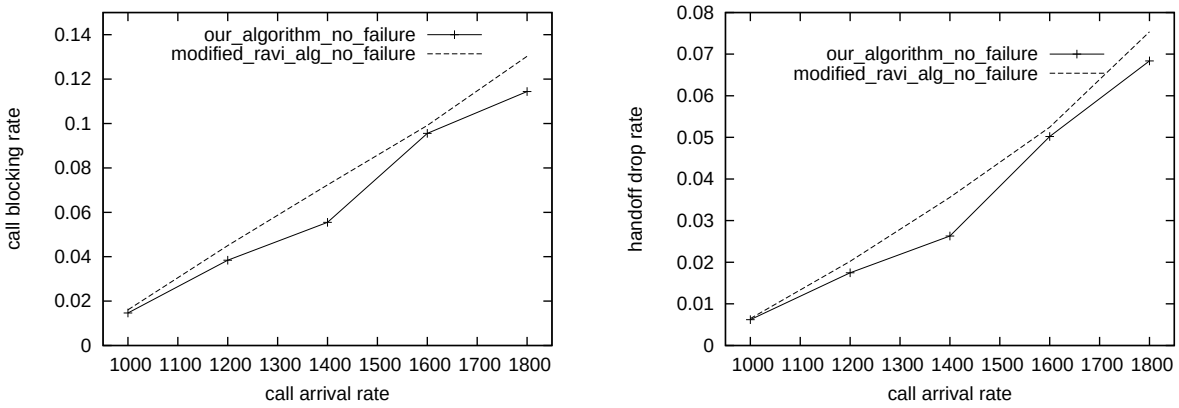
Mean call arrival rate in a normal cell	λ
Mean call arrival rate in a hot cell	3λ
Mean inter-handoff rate in a normal cell	$1/80s$
Mean inter-handoff rate in a hot cell	$1/180s$
Mean rate changing from normal to hot state	$1/1800s$
Mean rate changing from hot to normal state	$1/180s$
Mean service time per call	$180s$

2.5.3 Simulation Results

In the simulation, each MH generates one or more calls, including new calls and inter-handoff calls. To remove the start-up transients, simulation data was collected only after the first $10,000$ calls were processed. The simulation ended after $100,000$ calls were processed. Data was retrieved at the end of the simulation and used to compute the various metrics. In the following figures, the x-axis represents call arrival rate. The y-axis represents call blocking rate, handoff drop rate, call failure rate, and average number of messages per call respectively. The simulation was conducted under two scenarios: without cell failures and with cell failures. We show the simulation results of these two scenarios below.

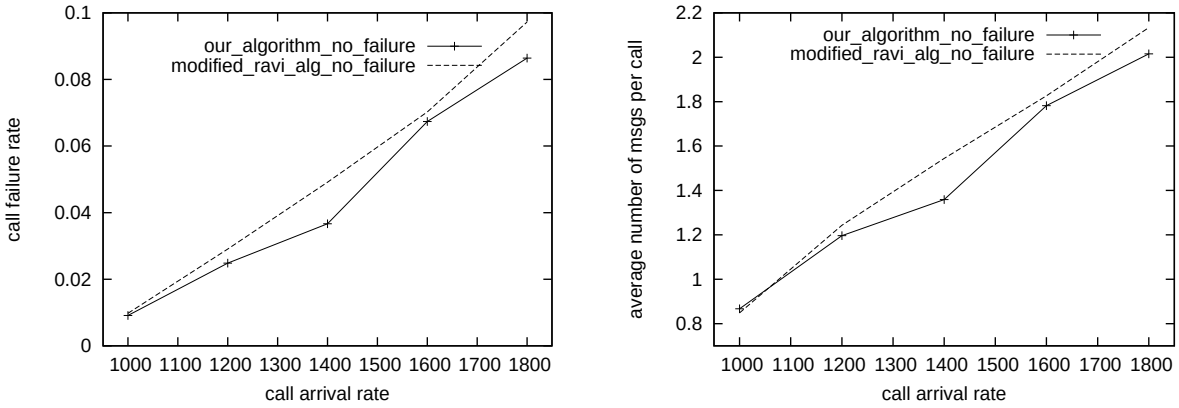
- **Without Cell Failures:** We compare the performance of our algorithm with the performance of the modified version of the algorithm proposed in [33] under the non-uniform traffic pattern and without cell failures. The comparison is made with respect to four metrics: call blocking rate, handoff drop rate, call failure rate, and message complexity. The simulation results are shown in Figure 2.7 and Figure 2.8. From these figures, we can see that all the metrics increase with the increase of call arrival rate. This is expected because when call arrival rate increases, there should be more call failures and the number of messages exchanged between MSSs for channel borrowing should be higher. As can be seen from the simulation results, with respect to all the four metrics, our algorithm performs better than the modified version of the algorithm in [33].

- **With Cell Failures:** Since the algorithm proposed in [33] is not fault tolerant, we did not simulate their algorithm with cell failures. Here, we only evaluated the performance of our



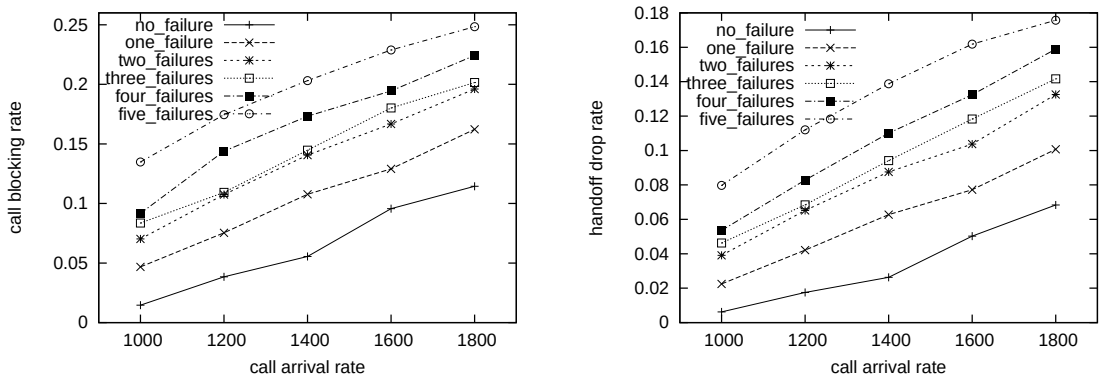
(a) (b)

Figure 2.7: Performance without cell failure (a) call blocking rate (b) handoff drop rate



(a) (b)

Figure 2.8: Performance without cell failure (a) call failure rate (b) message complexity

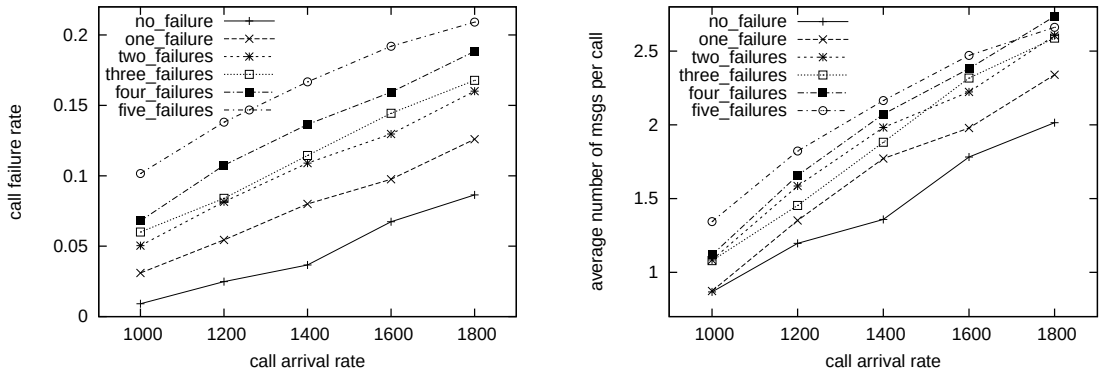


(a) (b)

Figure 2.9: Performance with cell failure (a) call blocking rate (b) handoff drop rate

algorithm with cell failures and showed the experimental results. In our simulation, we set the number of cell failures from 0 to 5. The simulation was run under exactly the same scenario, except that each time the number of cell failures is different. We evaluated the performance of our algorithm with respect to the number of cell failures and call arrival rate. The simulation results are shown in Figure 2.9 and Figure 2.10. As can be seen from these figures, all the four metrics increase with the increase of call arrival rate and the number of cell failures. When there is no cell failure and when call arrival rate is 1800, call blocking rate, handoff drop rate, call failure rate, and the average number of messages per call are 11.4%, 6.8%, 8.6% and 2.015 respectively. With the same call arrival rate, when the number of cell failures increases to five, the four metrics are 24.8%, 17.6%, 20.9%, and 2.66 respectively, which shows an approximately 11% to 13% increase in call failure (blocking and handoff-drop) rate over the case without cell failures. In terms of message complexity, we only observe an increase of 0.645. The maximum number of cell failures in our simulation is five, which is 13.89% of all the cells in the system (we have 36 cells total). It follows that in terms of call failure rate, when 13.89% of cells in the system fail, our algorithm can still support about 79.1% of the calls when the system has a very heavy load (when call arrival rate is 1800). Thus, our algorithm is fault tolerant and performs well even when there are up to 13.89% cells in the system fail and when the traffic load is heavy.

We observed that our algorithm has one limitation. At a cell C_i, if the number of *reply* messages is less than two, or if the neighbors from which C_i has received a *reply* message do not form any



(a) (b)

Figure 2.10: Performance with cell failure (a) call failure rate (b) message complexity

group, then our algorithm will drop the call. Thus, in this case, our algorithm performs the same as the algorithm proposed in [33].

2.6 Conclusion

In this chapter, we proposed a fault-tolerant channel allocation algorithm which achieves high channel utilization. In our algorithm, in order to borrow a channel, a cell does not need to receive channel usage information from all its neighbors, which makes the algorithm fault-tolerant. A cell may borrow a channel successfully even based on some partial channel usage information it receives from some of its neighbors. Moreover, a cell can lend a channel to multiple borrowers (at most 3) concurrently as long as no two of them are neighbors. So, our approach achieves high channel utilization. Our performance evaluation supports the fact that our algorithm is fault-tolerant and has low call drop rate.

Chapter 3

A Fault-Tolerant Channel Allocation Algorithm for Cellular Networks with Mobile Base Stations

3.1 Introduction

In traditional cellular networks, the entire geographical area covered by a cellular network is divided into smaller regions, called *cells*. Each cell contains a fixed *Base Station* (*BS*) serving *Mobile Hosts* (*MH*) [33] in that cell. The link between an MH and a BS is wireless, while the link between BSs is wired in general.

Cellular networks with Mobile Base Stations (*MBS*) differ from traditional cellular networks in the following aspects:

- Base stations move in cellular networks with MBSs. Each MBS is responsible for allocating channels for MHs in its cell. The geographical area covered by an MBS changes dynamically as the MBS moves. The neighborhood relationship of an MBS is not fixed; it changes dynamically due to the mobility of MBSs. Therefore, channel allocation algorithms proposed for traditional cellular networks assuming a static neighborhood relationship are not applicable in this situation.

- The links between MBSs are also wireless. So, channels will need to be allocated for supporting communication between MBSs. The wireless links between MBSs are referred to as *inter-cell communication links*, while the wireless links between an MH and the MBS in its cell are called *intra-cell communication links*.

- Co-channel interference may arise when two MBSs, which are not interference neighbors initially, move into each other's neighborhood and they happen to use the same channel concurrently. In this case, the two MBSs need to cooperate with each other and solve the co-channel interference.

An example of cellular networks with MBSs is shown in Figure 3.1.

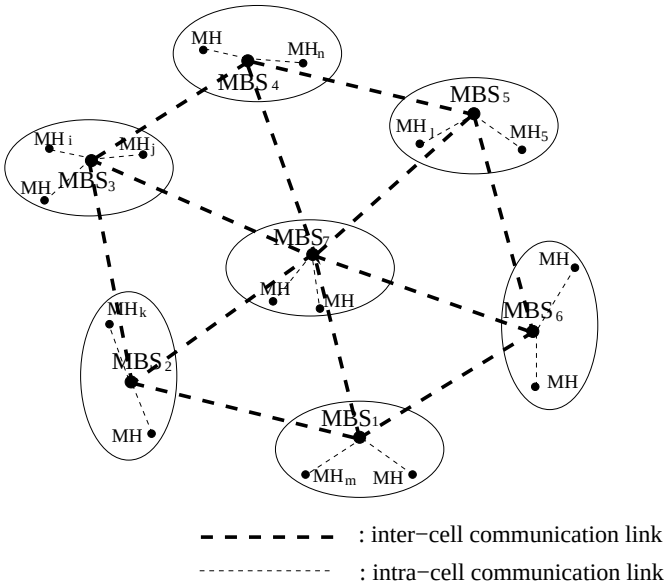


Figure 3.1: Cellular networks with mobile base stations

Throughout this chapter, we use the terms "cell" and "the MBS in the cell" interchangeably.

3.1.1 Motivation

Although considerable work has been done in channel allocation for traditional cellular networks with static base stations, not much attention has been paid to solve the channel allocation problem for cellular networks with mobile base stations. Cellular networks with mobile base stations are suitable for battlefield environment. In such an environment, base stations can be mounted on tanks, whereas MHs (such as PDAs) may be carried by soldiers. Another application of this type of network is for disaster recovery, where base stations are mounted on vehicles and rescuers are equipped with mobile hosts.

Due to the mobility of base stations, channel allocation approaches proposed for traditional cellular networks will not work for cellular networks with MBSs. New algorithms need to be developed to solve the channel allocation problem in such networks.

3.1.2 Contribution

In this chapter, we extend the channel allocation algorithm proposed in Chapter 2 to suit for cellular networks with mobile base stations. The proposed algorithm has the following desirable features:

- **Fault-Tolerance:** Base stations are susceptible to failure; this is more likely in an area such as a battle field, where the environment is hostile. When this happens, the performance of the system will degrade if the channel allocation algorithm does not take failure into consideration. Our algorithm is fault-tolerant in the sense that in order to borrow a channel, a base station does not need to know the channel usage information of all neighboring base stations. This feature enables our algorithm to work reasonably well in the presence of failures of base stations.

- **Efficient Channel Utilization:** Under the proposed algorithm, a cell can lend a channel to more than one cell concurrently as long as no two of these cells are neighbors, thus increasing the channel utilization.

The rest of the chapter is organized as follows. System model is given in Section 3.2. Related works are reviewed in Section 3.3. In Section 3.4 we present our algorithm. Correctness of the algorithm is proved in Section 3.5. Our algorithm is compared with related work in Section 3.6, followed by simulation results in Section 3.7. Section 3.8 concludes the chapter.

3.2 System Model

In cellular networks with MBSs, channels need to be allocated to support both intra-cell communication (i.e., communication between an MH and an MBS) and inter-cell communication (i.e., communication between two MBSs). MBSs share the responsibility to allocate channels and ensure that no co-channel interference arises. To simplify the channel allocation algorithm, the set of channels in the system is divided into two disjoint subsets, one to support intra-cell communication, the other to support inter-cell communication [29].

An MBS serves a cell with radius d, i.e., can support an intra-cell communication between an MH and itself if the distance between them is less than d. If a channel is being used to

support an intra-cell communication by an MBS, say MBS_i, then the same channel cannot be used concurrently within a radius of $k_1 \times d$ ($k_1 > 1$), with center at MBS_i. This range is called *interference range* of MBS_i. This is shown in Figure 3.2.

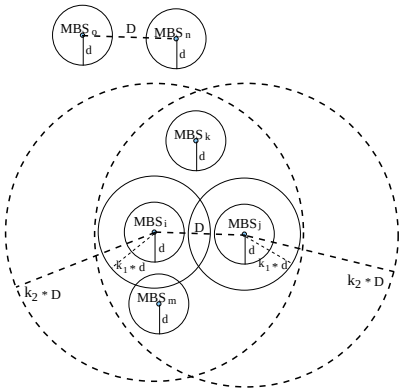


Figure 3.2: Neighborhood of an MBS

An MBS, whose cell intersects with the interference range of MBS_i, is called an *intra-neighbor* of MBS_i, and the set of all intra-neighbors of MBS_i is denoted by NB_{intra_i}. If a channel r is being used by MBS_i for an intra-cell communication, then none of its intra-neighbors can use r concurrently. But r can be used at the same time by an MBS which is not an intra-neighbor of MBS_i. For example, in Figure 3.2, MBS_m is an intra-neighbor of MBS_i, while MBS_j and MBS_k are not intra-neighbors of MBS_i. If a channel r is being used by MBS_i, then MBS_m cannot use the same channel r concurrently, but r can be used concurrently by MBS_j or MBS_k.

When two MBSs need to communicate with each other, they also need a channel to support their communication since the link between them is wireless too. An inter-cell communication can be established between two MBSs only if the distance between them is no greater than a threshold distance, denoted by D. If two MBSs, MBS_i and MBS_j, communicate with each other using a channel r, then r cannot be used concurrently by any other MBS, say MBS_x, if the distance between MBS_x and any of the two MBSs (MBS_i and MBS_j) is less than a threshold value: $k_2 \times D$ ($k_2 > 1$) [1]. An MBS is called an *inter-neighbor* of MBS_i if the distance between them is less than $k_2 \times D$, and the set of all inter-neighbors of MBS_i is denoted by NB_{inter_i}. For example, in Figure 3.2, the distance between MBS_i and MBS_j is D. If a channel r is being used

[1] $D > d$ because the power level that an MBS uses to communicate with another MBS is usually greater than that used to reach an MH in its cell; therefore, the transmission range that an MBS can reach in an inter-cell communication is larger. In addition, $k_2 \times D > k_1 \times d$.

to support the inter-cell communication between MBS_i and MBS_j, then r cannot be used by any inter-neighbor of either MBS_i or MBS_j. In this case, MBS_k and MBS_m are inter-neighbors of both MBS_i and MBS_j, and hence they cannot use r. But r can be used by MBS_n and MBS_o since neither of them is an inter-neighbor of MBS_i or MBS_j.

We assume that an MBS can exchange message with any neighbor (either an intra-neighbor or an inter-neighbor) by transmitting signals at a power level high enough to reach the neighbor. We also assume that an MBS transmits its own beacon signal periodically and an MBS has the knowledge of the identity of its neighbors by listening to their beacons [29]. Thus if a new MBS, say MBS_j, moves into the neighborhood of MBS_i, MBS_i can know this new neighbor's identity by listening to its beacons, and MBS_i will add this new neighbor MBS_j to the set of its neighbors. If an MBS, say MBS_i, has not received a neighbor MBS_j's beacon signal for some fixed period of time, then MBS_i assumes that either MBS_j moves out of its interference neighborhood or MBS_j has crashed, and removes MBS_j from the set of its neighbors.

In general, we assume that an MH that stays in a cell is likely to move with the MBS in the same cell for some period of time. After that period of time, it may move to other cells. If an MH involved in a communication moves from one cell to another, then the channel being used for the communication should be released to the cell from which it moves and a new channel should be allocated by the cell to which it moves. This is referred to as *inter-handoff* or *handoff*. Note that channels for inter-cell communication may also be needed due to the movement of an MH. For example, in Figure 3.1, suppose MH_i moves from the cell of MBS_3 to the cell of MBS_2 during its communication with MH_j in the cell of MBS_3. In this case, not only a channel for an intra-cell communication needs to be allocated by MBS_2, but also channels for inter-cell communication need to be allocated for the link between MBS_3 and MBS_2 or through other MBSs.

3.3 Related Works

In [34], the authors propose a distributed channel allocation algorithm for cellular networks with mobile base stations. Channels are allocated to support communication between mobile base stations (referred to as backbone links) and also communication between mobile base stations and mobile hosts (referred to as short-hop links). When allocating channels to support communication,

no distinction is made between channels used for backbone links and short-hop links. A mobile base station's neighborhood is divided into three regions: *no-use region*, *partial-use region*, and *full-use region*. If a channel r is used by a mobile base station, say MBS_i, then r cannot be used concurrently by any other mobile base station, say MBS_j, which is in the no-use region of MBS_i. However, r can be used concurrently by a mobile base station, say MBS_k, to support a short-hop link, if MBS_k is in the partial-use region of MBS_i. If a mobile base station, say MBS_l, is in the full-use region of MBS_i, then it can use r concurrently to support a short-hop link. Moreover, it can use r to support a backbone link concurrently between itself and another mobile base station, say MBS_m, if MBS_m is also in the full-use region of MBS_i. When allocating channels, a mobile base station may need to take into account neighbors in some or all of the regions. This algorithm is not fault-tolerant in the sense that a mobile base station needs to get channel usage information from each neighbor to borrow a channel.

In [29, 30], the authors propose a distributed dynamic channel allocation algorithm for cellular networks with mobile base stations. In this algorithm, the set of all channels in the system is divided into two disjoint subsets: one for *short-hop links* (i.e., to support communication between an MBS and an MH in its cell); the other for *backbone links* (i.e., to support communication between MBSs). The algorithm consists of two parts.

- **short-hop channel allocation:** When an MBS, say MBS_i, needs a channel, it first checks whether there exists an available channel allocated to it. If there exists such a channel, it can use this channel. Otherwise, it sends a request message to all neighboring MBSs within the short-hop channel reuse distance asking for their channel usage information. Upon receiving replies including such information from neighbors, it computes the set of channels which can be borrowed. It selects a channel r from this set (if there is any) and consults with its neighbors to which r has been allocated on whether it can borrow this channel to use. It can use the selected channel if all the neighbors it consults grant its request.

- **backbone channel allocation:** Whenever an MBS, say MBS_i, wants to communicate with another MBS, say MBS_j, all the base stations within backbone channel reuse distance of either MBS_i or MBS_j are polled to gather their channel usage information. A channel is chosen to support the communication if the channel is not being used by MBS_i, MBS_j

and the base stations that are polled. When the communication between MBS_i and MBS_j terminates, the channel supporting this communication is released to the system.

In their [29, 30] short-hop channel allocation algorithm, when an MBS does not receive a reply from a neighbor within a timeout period, it assumes that the neighbor either has crashed or moved out of its co-channel interference range. This assumption may not necessarily be true because messages can be lost. Such an assumption may lead to co-channel interference. Moreover, in their short-hop channel allocation algorithm, an MBS lends a channel to at most one MBS. This restricts channel reuse because an MBS should be able to lend the same channel to more than one neighbor as long as using the same channel causes no co-channel interference among those neighbors.

Next, we present our channel allocation algorithm for intra-cell communication which allows a cell to lend a channel to multiple cells concurrently. Moreover, it can tolerate failures.

3.4 A Fault-Tolerant Distributed Channel Allocation Algorithm for Intra-Cell Communication

In this section, first we present the basic idea behind our algorithm, and then we present a detailed description of the algorithm for intra-cell communication. To support inter-cell communications, any channel allocation algorithm for inter-cell communications, such as the backbone channel allocation algorithm proposed in [29, 30], can be used. We assume that there are total N channels available in the system for intra-cell communication. Channels are totally ordered according to their frequency band, that is, the channel with the lowest frequency band is the first channel and the channel with the highest frequency band is the N^{th} channel [32]. Each message is timestamped with Lamport timestamp [23], the greater the timestamp, the lower the priority. Obsolete messages can be detected by comparing timestamps and discarded. In the following, we use cell the terms "cell C_i" and "the base station MBS_i in cell C_i" interchangeably. In the context of channel allocation for intra-cell communication, the term neighbor refers to an intra-neighbor.

3.4.1 Basic Idea

When a cell C_i needs a channel for an intra-cell communication, it picks an unused channel r from the set of channels allocated to it for use. If no such channel exists, it sends a request message to all

neighbors in NB_{intra_i}, asking for their channel usage information and neighborhood information. In this case, C_i is called a *borrower* and is said to be in *Search Mode*. When a cell C_j receives a request message from C_i, it sends a reply message to C_i if it has no outstanding request or its request has a greater timestamp than that of C_i's request. Otherwise, it defers sending a reply message to C_i. If C_j decides to defer C_i's request, it will send a reply message corresponding to C_i's request after it finishes its channel borrowing process. The reply message contains its channel usage information and its neighborhood information.

If C_i receives channel usage information from each neighbor, it computes the set of channels that have not been allocated to any of its neighbors or itself. If this set is not empty, then it picks a channel to use. Otherwise, it computes the set of channels which are allocated to some neighbors but not being used by any of them. If this set is not empty, then it picks a channel from this set and consults with those neighbors on whether it can use this channel. If this set is empty, then it drops the call.

Unlike in [29, 30], to borrow a channel, C_i does not need to get channel usage information from each of its neighbors. In case C_i gets channel usage information from only a subset of its neighbors, it checks for the following condition:

> For each neighbor C_j from which channel usage information has NOT been received by C_i, C_i has already received channel usage information from a neighbor of C_j.

If the above condition is satisfied, C_i divides the set of cells from which it received channel usage information into subsets (subsets may not necessarily be disjoint) in such a way that

- No two cells in the same subset are neighbors.
- For any neighbor C_j which is NOT in a subset, at least one of C_j's neighbors is in the subset.
- For any neighbor C_j from which channel usage information has NOT been received by C_i, a neighbor of C_j is in the subset.

For each subset constructed as described above, C_i computes the set of channels which are allocated to all the members of the subset but not being used by any of them. It takes a channel r from the computed set for a given subset of cells, and sends a transfer(r) message to all neighbors to which

r had been allocated (i.e., all members of the subset). This transfer(r) message contains the set of C_i's neighbors, NB_{intra_i}. When a cell C_j receives such transfer(r) message, its response depends on the current status of channel r. The response can be either an agree(r) or a refuse(r) message.

After C_i receives a response from each member of this subset, it borrows channel r successfully if all the responses are agree(r) messages. C_i notifies the neighbors to which channel r has been allocated about the result of its attempt to borrow r. Upon receiving such notice, a neighbor updates the status of r appropriately. The basic idea can be illustrated using Figure 3.3. In

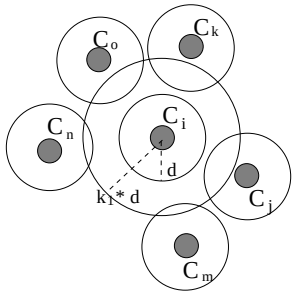


Figure 3.3: Illustration of the basic idea

Figure 3.3, $NB_{intra_i} = \{C_j, C_k, C_m, C_n, C_o\}$. Cell C_o is a common neighbor of C_n and C_k, and cell C_m and C_j are neighbors. C_j is not a neighbor of C_k, C_n or C_o; neither is C_m. C_k and C_n are not neighbors. Suppose C_i sends a request message for their channel usage information to each of its neighbors, but only receives reply messages from C_m, C_k and C_n. For C_o whose reply has not been received by C_i, C_i has already received a reply from C_k which is a neighbor of C_o. Similarly, for C_j, C_i has already received a reply from a neighbor of C_j, namely C_m. Thus, the condition mentioned earlier is satisfied. Hence, C_i divides the set of cells from which replies have been received into subsets according to the rules mentioned above. In this example, one such subset is $\{C_m, C_k, C_n\}$. If there is a channel r that has been allocated to all these three cells but is not being used by any of them, then C_i may borrow r. If all these three cells grant C_i's request for borrowing channel r, then C_i can use r safely, and no co-channel interference arises. All these three cells do not use r concurrently since they grant C_i's request for borrowing r, and channel r is not allocated to any other neighbor of C_i. If a neighbor of C_i also tries to borrow r, then our algorithm guarantees that it cannot acquire r concurrently (we prove this in Section 3.5). Moreover, a cell is allowed to lend the same channel to multiple cells as long as no two of them are neighbors. For example, in Figure 3.3, suppose both C_o and C_m want to borrow a channel r from C_i. Then, C_i can grant both

of their requests because they are not neighbors and using r by them will not cause co-channel interference.

In our algorithm, we assume that when an MBS moves, it updates the set of channels allocated to it based on the set of channels allocated to its new neighbors so that no co-channel interference arises. This can be done in the following way: when an MBS, say MBS_i, detects a new neighbor MBS_j, it sends the set of channels allocated to it and the number of its available channels to MBS_j, asking for the same information from MBS_j. Upon receiving such a message, if MBS_j has not sent this information to MBS_i yet, it sends this information to MBS_i. Upon receiving each other's information about the set of allocated channels, both MBS_i and MBS_j update their sets of allocated channels appropriately. If they happen to have the same channel r allocated to both of them, then the one with fewer available channels keeps the channel, while the other removes the channel from its set of allocated channels. Ties can be broken by comparing the identities of MBSs. This is true for both intra-cell communications and inter-cell communications. For example, for intra-cell communications, if two MBSs, which initially are not neighbors that use the same channel r, move into each other's neighborhood, co-channel interference arises (note that we assume that an MH normally moves with the MBS in the same cell). To avoid further interference, at least one of them has to switch to a different channel. We assume that this will be done by the MBS with more available channels. Similar things can be done for inter-cell communications. If two pairs of MBSs, which initially use the same channel without interference, move in such a way that interference arises, then at least one of them has to switch to a different channel. **From now on, we concentrate on the channel allocation for intra-cell communications only.**

3.4.2 Data Structures

Each cell C_i maintains the data structures shown in Table 3.1.

Initially, $Allo_{intra_i}$, $Busy_{intra_i}$, $Tran_{intra_i}$, $Grant_{intra_i}(r)$ and $Lent_{intra_i}(r)$ are all empty. $\forall\, r \in Allo_{intra_i}$, $Grant_{intra_i}(r)$ is the set of cells to which C_i has sent an *agree(r)* message, and from which C_i has not received a *release(r)* message or a *keep(r)* message. C_i lends channel r to a neighbor C_j only when it is notified by C_j to release r (i.e., C_i receives a *release(r)* message from C_j). $Lent_{intra_i}(r)$ is the set of cells from which C_i has received a *release(r)* message. That is,

Table 3.1: Data Structures

NB_{intra_i}:	the set of intra-neighbors of C_i.
$Intra$:	the set of channels for intra-cell communication in the system.
$Allo_{intra_i}$:	the set of channels allocated to C_i for intra-cell communication.
$Busy_{intra_i}$:	the set of channels being used by C_i for intra-cell communication.
$Tran_{intra_i}$:	the set of channels marked for transfer by C_i for intra-cell communication.
$Grant_{intra_i}$:	a set of sets maintained by C_i.
$Lent_{intra_i}$:	a set of sets maintained by C_i.

$Lent_{intra_i}(r)$ denotes the set of cells to which C_i has lent channel r. Under all circumstances, $Busy_{intra_i} \subseteq Allo_{intra_i}$, $Tran_{intra_i} \subseteq Allo_{intra_i}$, and $Busy_{intra_i} \cap Tran_{intra_i}$ is an empty set.

3.4.3 A Channel Allocation Algorithm for Intra-cell Communication

Formal description of our channel allocation algorithm is presented in Table 3.2.

3.5 Correctness of the Algorithm

Lemma 3.5.1 *Two neighboring cells, C_i and C_j, are not allowed to borrow the same channel concurrently under the proposed algorithm.*

Proof: Let r_1 be the channel borrowed by C_i, r_2 be the channel borrowed by C_j. Without loss of generality, we assume that C_i's request has lower priority than C_j's request. We prove that $r_1 \neq r_2$. Since C_j's request has higher priority, C_j will defer sending a reply to C_i until it finishes its channel borrowing process. After C_j borrows r_2, it adds r_2 to both $Allo_{intra_j}$ and $Busy_{intra_j}$. Then, it sends a reply message to C_i. The following two cases arise for C_i.

- case 1: C_i borrows r_1 as a result of getting reply messages from all of its neighbors for its request. If $r_1 \in F_{intra_i} := Intra - \cup Allo_{intra_k}$ (step C-(1)-α of the algorithm), then $r_1 \neq r_2$, because $r_2 \in Allo_{intra_j}$, and C_j is a neighbor of C_i. If $r_1 \in F_{intra_i} := Intra - \cup Busy_{intra_k}$ (step C-(1)-β of the algorithm), then $r_1 \neq r_2$, because $r_2 \in Busy_{intra_j}$, and C_j is a neighbor of C_i.

- case 2: C_i borrows r_1 as a result of getting reply messages from only a subset of its neighbors for its request. In this case, C_i must have borrowed r_1 from some subset S_I such that

Table 3.2: The Intra-Cell Channel Allocation Algorithm

A: When a cell C_i needs a channel to support a call,

If $(F_{intra_i} := Allo_{intra_i} - Busy_{intra_i} - Tran_{intra_i}) \neq \emptyset$, it picks a channel $r \in F_{intra_i}$ with the highest order to support the call. $Busy_{intra_i} := Busy_{intra_i} \cup \{r\}$. When the call terminates, $Busy_{intra_i} := Busy_{intra_i} - \{r\}$.

If $F_{intra_i} = \emptyset$, it sets a timer and sends a *request* message to each cell in NB_{intra_i}.

B: When a cell C_j receives a *request* message from cell C_i,

If its own *request* has a lower timestamp than C_i's *request*, then it defers sending a *reply* message to C_i.

Otherwise, it sends to C_i a $reply(Allo_{intra_j}, Busy_{intra_j}, NB_{intra_j})$ message.

C: When C_i gets *reply* messages from all neighbors or the timer expires, it sets a new timer and does the following:

(1): If it has received *reply* messages from all of its neighbors,

α: If $(F_{intra_i} := Intra - \cup Allo_{intra_k}) \neq \emptyset$ $(k \in NB_{intra_i} \cup \{i\})$, it picks $r \in F_{intra_i}$ with the highest order to support the call. $Allo_{intra_i} := Allo_{intra_i} \cup \{r\}$, $Busy_{intra_i} := Busy_{intra_i} \cup \{r\}$. $Grant_{intra_i}(r) := \emptyset$, $Lent_{intra_i}(r) := \emptyset$.
β: Else if $(F_{intra_i} := Intra - \cup Busy_{intra_k}) \neq \emptyset$ $(k \in NB_{intra_i} \cup \{i\})$, it picks a channel $r \in F_{intra_i}$ with the lowest order for transfer and sends a *transfer(r)* message to each neighbor to which r has been allocated. NB_{intra_i} is included in the *transfer(r)* message.
γ: Else if $F_{intra_i} = \emptyset$, then it drops the call.

(2): Else, let $Received_{intra_i} := \{C_j : C_i$ *has received a reply message from* C_j.$\}$.

α: If $\forall C_k \in (NB_{intra_i} - Received_{intra_i})$, $\exists C_m$ *such that* $C_m \in NB_{intra_k} \cap Received_{intra_i}$, then C_i divides $Received_{intra_i}$ into subsets according to the following rules:

(i) No two cells in the same subset are neighbors.
(ii) $\forall C_n \in NB_{intra_i}$ that is not in a subset, $\exists C_o \in NB_{intra_n} \cap Received_{intra_i}$ in the subset.
(iii) $\forall C_p \in NB_{intra_i} - Received_{intra_i}$, each subset includes some cell, C_q, such that $C_q \in NB_{intra_p} \cap Received_{intra_i}$.

Let the subsets be: $S_1, \ldots, S_K$.
$\forall\ S_I\ (I \in \{1, \ldots, K\}), F_{intra_i}(I) := \cap_{C_n \in S_I}(Allo_{intra_n} - Busy_{intra_n})$. If $\forall I \in \{1, \ldots, K\}$, $F_{intra_i}(I) = \emptyset$, then C_i drops the call; otherwise, let $F_{intra_i} := \cup_{I \in \{1,\ldots,K\}} F_{intra_i}(I)$. C_i picks a channel r with the lowest order from set $F_{intra_i}(I)$ such that $F_{intra_i}(I) \neq \emptyset$ and $|F_{intra_i}(I)|$ is maximal. It sends to each cell in S_I a *transfer(r)* message with NB_{intra_i} attached.
β: Else it drops the call.

D: When cell C_j receives a *transfer(r)* message from cell C_i,

If $r \notin Busy_{intra_j} \wedge (\forall C_m \in (Grant_{intra_j} \cup Lent_{intra_j}), C_m \notin NB_{intra_i})$, it sends an *agree(r)* message to C_i and $Grant_{intra_j}(r) := Grant_{intra_j} \cup \{C_i\}$. If $r \in Allo_{intra_j}$ and $r \notin Tran_{intra_j}$, then $Tran_{intra_j} := Tran_{intra_j} \cup \{r\}$.

Otherwise, it sends a *refuse(r)* message to C_i.

E: When C_i receives a response to each of its *transfer(r)* message and the timer set in step C does not expire, it cancels the timer and does the following:

1: If all of them are *agree(r)* messages, then it uses r to support the call. $Allo_{intra_i} := Allo_{intra_i} \cup \{r\}$ and $Busy_{intra_i} := Busy_{intra_i} \cup \{r\}$. $Grant_{intra_i}(r) := \emptyset$ and $Lent_{intra_i}(r) := \emptyset$. It sends a *release(r)* message to each cell from which an *agree(r)* message has been received.
2: If not all of them are *agree(r)* messages, it sends a *keep(r)* message to each cell from which an *agree(r)* message has been received. Channel r is removed from the set F_{intra_i}. C_i tries to borrow another channel in F_{intra_i} if possible. If C_i cannot borrow a channel successfully before the timer set in step C expires, then it drops the call.

F: When cell C_j receives a *release(r)* message from cell C_i,

It sets $Grant_{intra_j}(r) := Grant_{intra_j}(r) - \{C_i\}$ and $Lent_{intra_j}(r) := Lent_{intra_j}(r) \cup \{C_i\}$. If $r \in Allo_{intra_j}$, then it sets $Allo_{intra_j} := Allo_{intra_j} - \{r\}$ and $Tran_{intra_j} := Tran_{intra_j} - \{r\}$.

G: When cell C_j receives a *keep(r)* message from cell C_i,

It sets $Grant_{intra_j}(r) := Grant_{intra_j}(r) - \{C_i\}$. If $r \in Allo_{intra_j}$ and $Grant_{intra_j}(r) = \emptyset$, then it sets $Tran_{intra_j} := Tran_{intra_j} - \{r\}$.

H: After cell C_i acquires a channel to support a call or it drops a call,

It sends a $reply(Allo_{intra_i}, Busy_{intra_i}, NB_{intra_i})$ message to all those *request* messages to which a *reply* has been deferred.

$F_{intra_i(I)} \neq \emptyset$, where $F_{intra_i(I)} := \cap_{C_n \in S_I}(Allo_{intra_n} - Busy_{intra_n})$ (step C-(2)-α of the algorithm). We have $r_1 \in F_{intra_i}(I)$. Therefore, $r_1 \notin Busy_{intra_n}$ for any $C_n \in S_I$. Following two sub-cases arise.

- sub-case 1: $C_j \in S_I$. Because C_j sends a reply message to C_i only after it finishes its channel borrowing process, $r_2 \in Busy_{intra_j}$. Therefore, $r_1 \neq r_2$ because $r_1 \notin Busy_{intra_j}$.
- sub-case 2: $C_j \notin S_I$. Then, $\exists C_k$ *such that* $C_k \in S_I \wedge C_k \in NB_{intra_j}$. Therefore, $r_1 \in Allo_{intra_k}$. We have $r_1 \neq r_2$ because $r_2 \notin Allo_{intra_k}$ (since $r_2 \in Allo_{intra_j}$ and C_k is a neighbor of C_j).

Thus, two neighboring cells are not allowed to borrow the same channel concurrently under the proposed algorithm. □

Lemma 3.5.2 *Under the proposed algorithm, two neighboring cells do not use the same channel concurrently for an intra-cell communication at any time.*

Proof: Initially, no channel is allocated to any cell. Since neighboring cells do not borrow the same channel concurrently, it follows that neighboring cells do not use the same channel concurrently. □

Lemma 3.5.3 *The proposed channel allocation algorithm is deadlock free.*

Proof: Under the proposed algorithm, when a cell sends a *request* message (or a *transfer(r)* message, where r is the channel selected to borrow), it sets a timer. A cell starts computing whenever its timer expires or it gets from each neighbor a response corresponding to each of its *request* messages (or *transfer(r)* messages). So, there is no hold and wait situation. Thus, the proposed algorithm is dead-lock free. □

3.6 Comparison to Related Works

In [34], the authors proposed a distributed algorithm for allocating channels for both intra-cell and inter-cell communication. We do not compare our algorithm with that in [34], because our algorithm focuses only on channel allocation for intra-cell communication.

To our knowledge, the algorithm proposed in [29, 30] is the only channel allocation algorithm in the literature for cellular networks with mobile base stations that divides the set of channels into two disjoint subsets for the two different types of links, namely, intra-cell and inter-cell links. So we compare our algorithm with their short-hop channel allocation algorithm, which we described in Section 3.3. Our channel allocation algorithm has the following advantages:

- **Fault-Tolerance**: Our algorithm is fault-tolerant because in order to borrow a channel, a cell does not need to get channel usage information from all its neighbors. In [29, 30], when a cell does not get channel usage information from a neighbor, it assumes that the neighbor either has moved out of its neighborhood or has crashed. This assumption may not be true. In cellular networks, messages exchanged between base stations could be lost. By making such an assumption, the algorithm in [29, 30] may cause co-channel interference (i.e., the algorithm may not work correctly). However, our algorithm does not rely on such an assumption.

- **No Co-Channel Interference**: Our algorithm ensures no co-channel interference for communication between a mobile host and a base station, while in the short-hop channel allocation algorithm proposed in [29, 30], co-channel interference may arise. In [29, 30], if a cell does not receive response from a neighbor within a timeout period, it assumes that either that neighbor has crashed or moved out of its co-channel interference range. But this assumption may not necessarily be true because messages could be lost. By assuming this, neighboring cells may choose the same channel to use concurrently. Thus, co-channel interference may arise. For example: in Figure 3.4, C_i's neighbors are: C_j, C_k, C_m and C_x. C_j's neighbors are: C_i, C_k, C_m and C_y. Any two of C_k, C_m, C_x and C_y are not neighbors. Suppose C_i needs to borrow a channel. It sends a *request* message to all neighbors and receives a *reply* message from all its neighbors. It chooses a channel r to borrow, where r is allocated to both C_j and C_x but not being used by any of them. It sends a *transfer(r)* message to them. Suppose that C_i receives an *agree(r)* message from C_x, but does not receive an *agree(r)* or *refuse(r)* message from C_j within a timeout period. According to the algorithm proposed in [29, 30], C_i assumes that C_j has crashed or moved out of its co-channel interference range and an *agree(r)* message is received from C_j. C_i uses channel r since it gets all *agree(r)* messages it

needs. But C_j may use channel r at the same time. A *refuse(r)* message may be sent by C_j, but this message is lost and C_i never receives such a message. Thus co-channel interference arises. Note that C_i may use r even when it does not receive a response from both C_x and C_j because of the assumption.

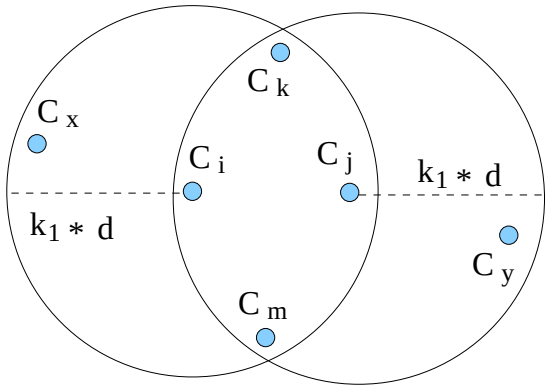


Figure 3.4: The cases where co-channel interference arises

3.7 Performance Evaluation

In this section, we present the results of the performance evaluation of our algorithm. We compare our algorithm with the short-hop channel allocation algorithm in [29, 30] for the reason stated in Section 3.6. The short-hop channel allocation algorithm in [29, 30] is not fault-tolerant, therefore, we cannot compare it with our algorithm directly. For purpose of comparison, we modified it to be fault-tolerant. The modification is as follows. When a cell C_i needs to borrow a channel from its neighbors, it sends a *request* message to each neighbor and sets a timer. If the timer times out before it receives replies from all neighbors, then it drops the call. Otherwise, it proceeds as normal. When a cell C_i sends a *transfer(r)* message to neighbors, it also sets a timer. If this timer expires before it can borrow a channel successfully, then it drops the call; otherwise, it uses the borrowed channel to support the call.

Three metrics are used to evaluate the performance of the algorithms: *call blocking rate*, *handoff drop rate*, and *call failure rate*. Call blocking rate is defined as the ratio of the number of new calls that cannot be supported (i.e., blocked new calls) to the total number of new calls. Handoff drop rate is defined as the ratio of the number of inter-handoff calls dropped to the total number of inter-handoff calls. Call failure rate is defined as the ratio of total number of calls

dropped (including blocked new calls and dropped inter-handoff calls) to the total number of calls processed. Call arrival rate is defined as the number of call requests per cell per hour.

3.7.1 Simulation Parameters

To evaluate the performance of the algorithm, we used CSIM18 [2] Simulation Engine to implement our simulation. CSIM18 is a process-oriented, discrete-event simulator. We assumed that a total of 300 channels are available for intra-cell communication. The simulated cellular network consists of 100 mobile base stations. Each mobile base station is associated with a number of mobile hosts. All the mobile base stations are within a square of 15 kilometers. At the beginning of the simulation, mobile base stations are distributed evenly within the square, so that adjacent stations are 1.5 kilometers apart either along x axis or y axis. The movement of a mobile base station is modeled as follows. Initially, each mobile base station chooses a destination within the square and a speed between 0 and a maximum speed of 10 kilometers per hour. Once a mobile base station selects a destination and a speed, it moves towards the destination with that speed until it arrives at the destination. Then, it stays there for a random period of time. Afterwards, it selects a new destination and a new speed and moves to the new destination. This movement pattern continues until the end of the simulation.

During the simulation, mobile hosts are generated according to call arrival rate at each cell covered by a mobile base station. Once generated, a mobile host sends a call request to the mobile base station and waits for the response. If a channel is allocated to the mobile host, then the mobile host uses this channel for communication. We assume that the average service time per call is 3 minutes. A mobile host stays with a mobile base station for an average of 30 minutes. After 30 minutes, it may move to an area covered by a neighboring mobile base station. If this happens while it is involved in a communication, an inter-handoff occurs.

We assume that the average one-way communication delay between two neighboring mobile base stations is 4 milliseconds. This average delay includes transmission delay, propagation delay, and message processing time. The value of the timers used in the algorithm is 8 milliseconds, which is twice the average one-way communication delay. Our algorithm only deals with the channel allocation problem for intra-cell communications.

[2]CSIM18 Simulation Engine is a product of Mesquite Software, Inc.

In our simulation, the transmission range of a mobile base station is 1.0 kilometer, and the intra-neighbor range of a mobile base station is 2.0 kilometers. Due to mobility, the neighborhood information of a mobile base station changes. Two mobile base stations, say MBS_i and MBS_j, which initially are not neighbors, may move towards each other and become neighbors, or two neighbors may move away from each other and cease to be neighbors. To track this change in neighborhood, we re-compute the neighborhood information for each mobile base station periodically. The re-compute period is set to be 120 seconds in the simulation. When two mobile base stations, MBS_i and MBS_j, that initially are not neighbors, move towards each other and become neighbors, co-channel interference may arise because some channels could have been allocated to both MBS_i and MBS_j. If this happens, the one with fewer available channels will keep such a channel, while the other will release the channel, in order to avoid co-channel interference.

The simulation is conducted under a non-uniform traffic pattern, which is more realistic. Under such a pattern, a cell can be in one of two states: normal state and hot state. When a cell is in normal state, call arrival rate is low, and when it is in hot state, call arrival rate is high. The parameters for non-uniform traffic pattern are given in Table 3.3 [6].

Table 3.3: Parameters for non-uniform traffic pattern

Mean call arrival rate in a normal cell	λ
Mean call arrival rate in a hot cell	3λ
Mean rate changing from normal to hot state	$1/1800s$
Mean rate changing from hot to normal state	$1/180s$
Mean service time per call	$180s$

3.7.2 Simulation Results

We simulated 72 different scenarios (8 different arrival rates, 9 different values of number of failed base stations, $8 \times 9 = 72$) for the two algorithms and ran each scenario ten times with different seeds. In the simulation, each MH generates one or more calls, including new calls and inter-handoff calls. To remove the start-up transients, data was collected only after the first $10,000$ calls were processed. The simulation ended after $100,000$ calls were processed. Data was retrieved at the end of the simulation and used to compute the metrics. To study the performance of the algorithm, we conducted the simulation over a wide range of call arrival rates. To test how well our

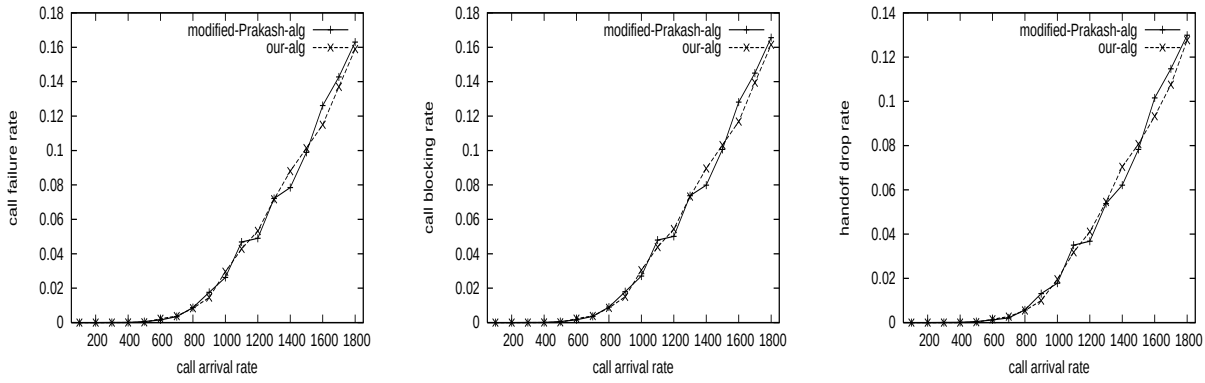


Figure 3.5: Performance without cell failure under non-uniform traffic pattern

algorithm tolerates failures, we conducted the simulation with different numbers of cell (mobile base station) failures. First, we show the simulation results of our algorithm without cell failures. Then, we analyze the performance of our algorithm with cell failures. In both cases, comparison to the modified version of the algorithm in [29, 30] is also presented.

1. **Without Cell Failures:** In this set of experiments, no cell fails. The simulation results are shown in Figure 3.5. As seen from the figure, all three metrics increase with call arrival rate. There is not much difference between the performance of the two algorithms. This is because when no cell fails, a cell, which sends out a request message to neighbors, can receive channel usage information from all its neighbors before its timer expires in most cases.

2. **With Cell Failures:** We evaluated the performance of our algorithm with cell failures and compared it with that of the modified version of the algorithm in [29, 30]. In our simulation, we used a total of 100 cells (i.e., mobile base stations). We conducted the simulation with the following numbers of cell failures: 1, 2, 3, 4, 5, 10, 15, and 20. The simulation was run under exactly the same scenario, except that each time the number of cell failures is different. The simulation results are shown in Figure 3.6 to Figure 3.13. As we can see from the figures, all three metrics, namely call blocking rate, handoff drop rate, and call failure rate, not only increase with call arrival rate, but also increase with the number of cell failures. This is reasonable because when some cell fails, then it is more difficult for a neighboring cell to find an available channel to borrow. The more cell failures, the harder for neighboring cells to borrow channels. In the presence of different numbers of cell failures shown in Figure 3.6

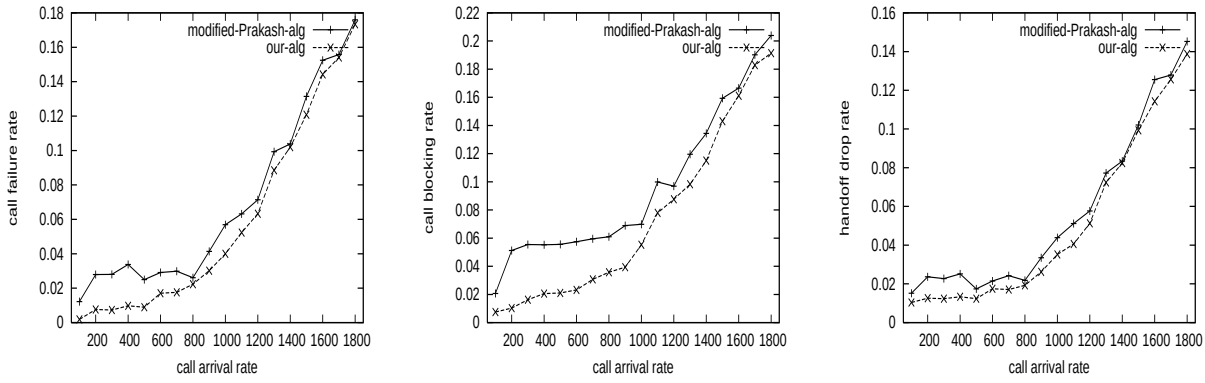


Figure 3.6: Performance with one cell failure under non-uniform traffic pattern

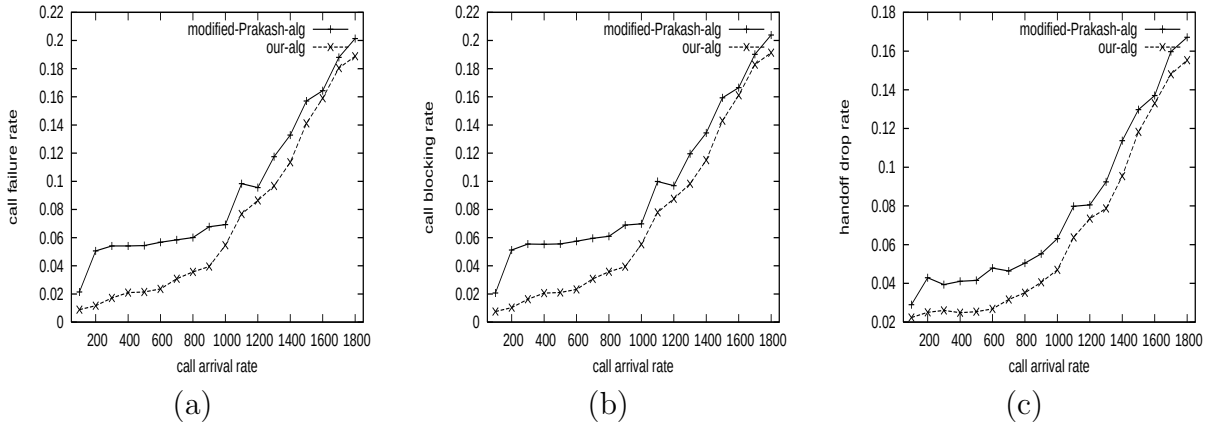
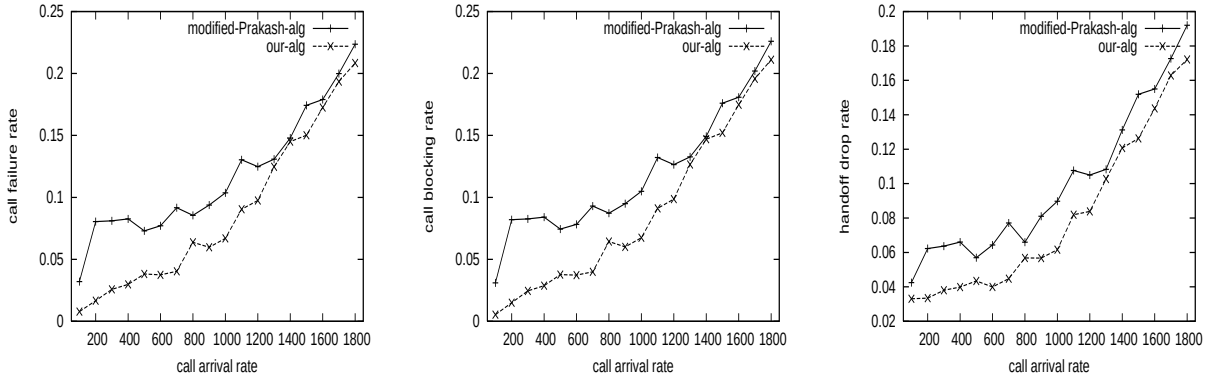


Figure 3.7: Performance with two cell failures under non-uniform traffic pattern

to Figure 3.13, our algorithm always has a lower call blocking rate, handoff drop rate, and call failure rate than the modified version of the algorithm in [29, 30]. The advantage of our algorithm becomes prominent with the increase in the number of cell failures, hence, our algorithm handles failures better.

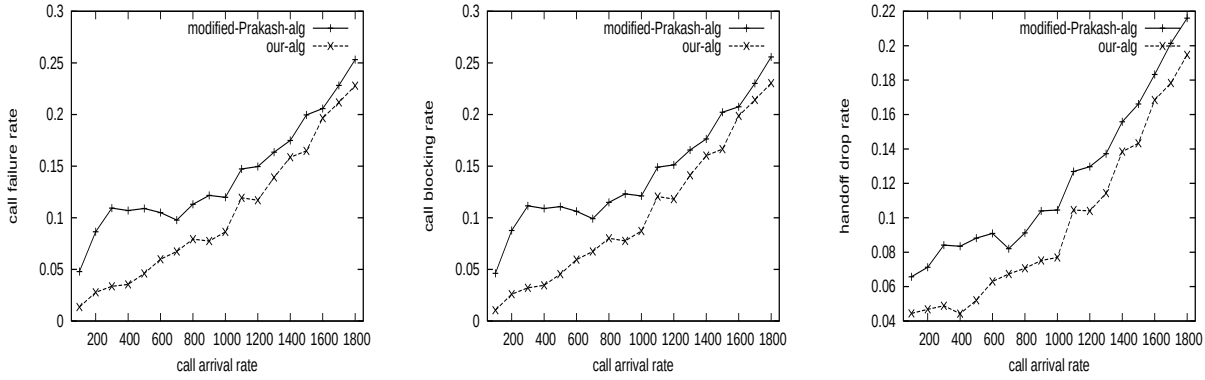
3.8 Conclusion

Although many channel allocation algorithms have been proposed for cellular networks with static base stations in the literature, not much work has been done for cellular networks with mobile base stations. To our knowledge, only two algorithms [29, 30] have been proposed in the literature to allocate channels for cellular networks with mobile base stations. However, both of these algorithms did not address fault-tolerance issue very well. We proposed a more efficient fault-tolerant channel allocation algorithm for commutation between a mobile host and a base station.



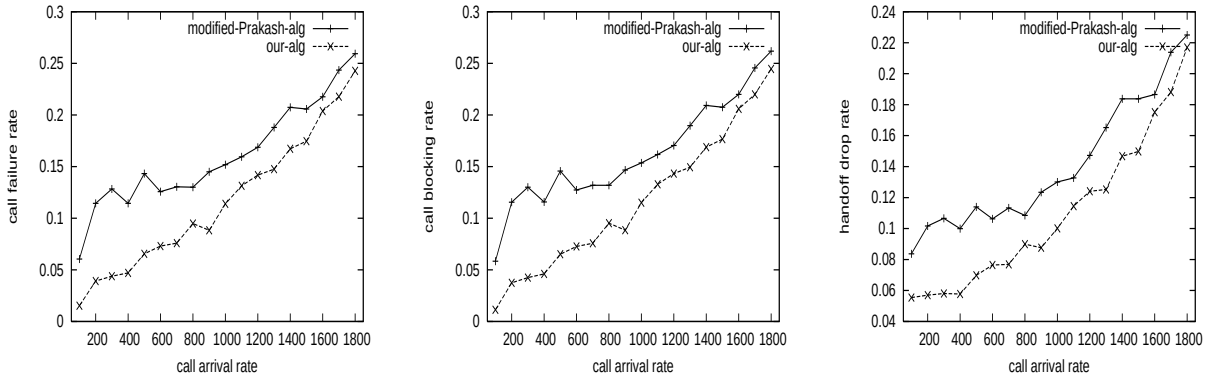
(a) (b) (c)

Figure 3.8: Performance with three cell failures under non-uniform traffic pattern



(a) (b) (c)

Figure 3.9: Performance with four cell failures under non-uniform traffic pattern



(a) (b) (c)

Figure 3.10: Performance with five cell failures under non-uniform traffic pattern

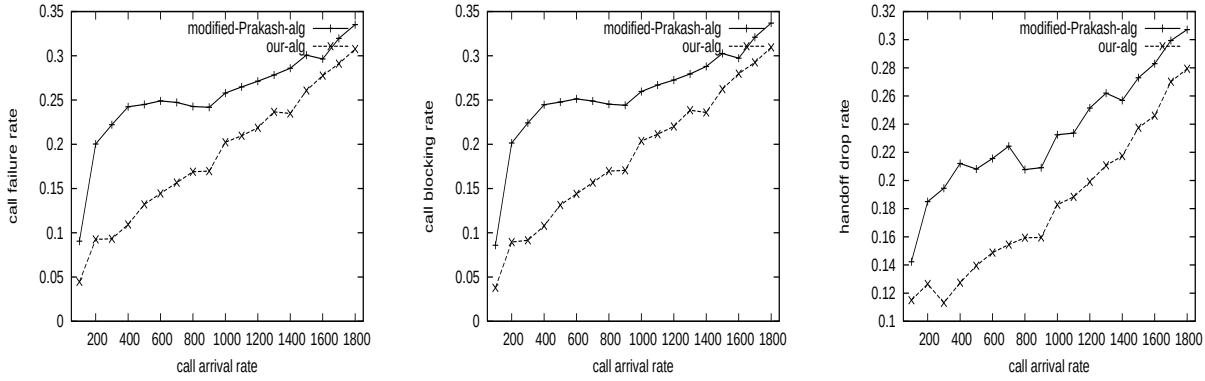


Figure 3.11: Performance with ten cell failures under non-uniform traffic pattern

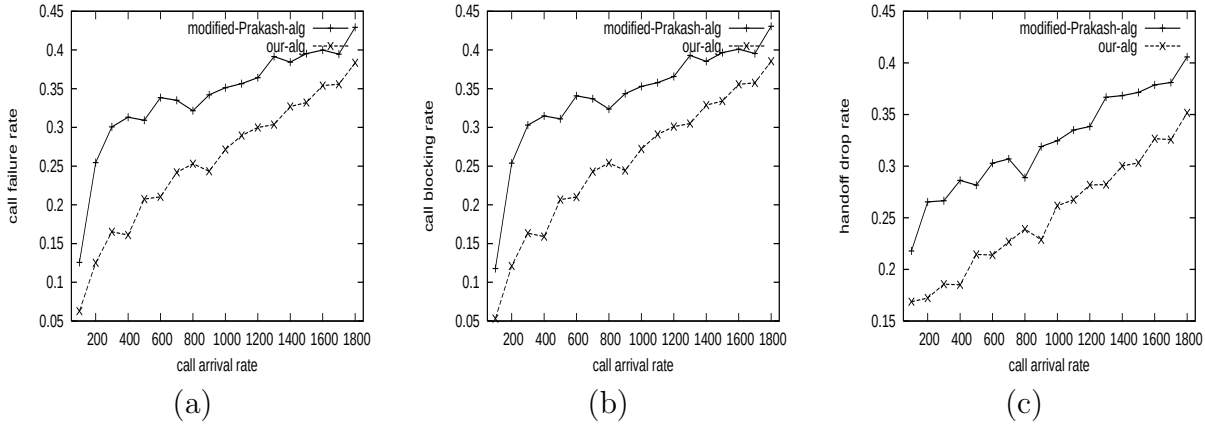


Figure 3.12: Performance with fifteen cell failures under non-uniform traffic pattern

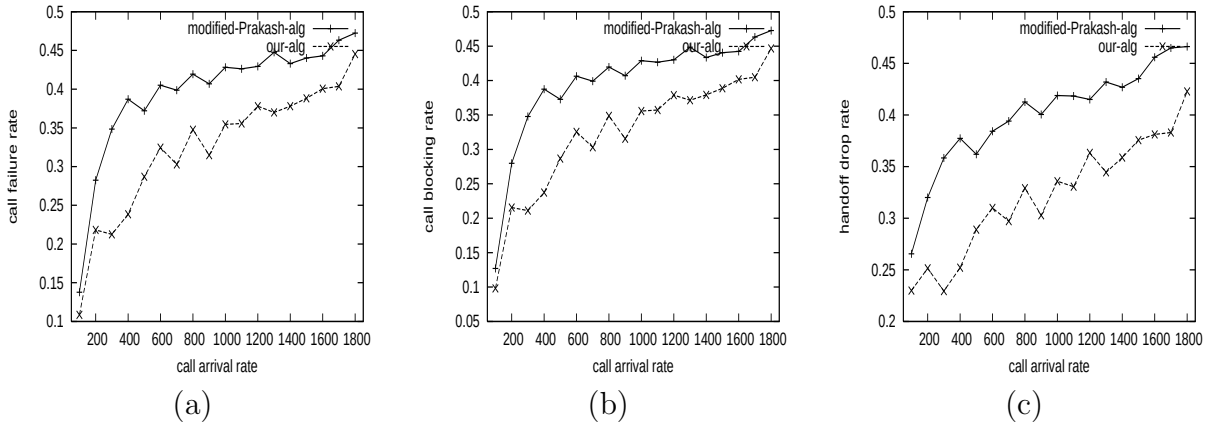


Figure 3.13: Performance with twenty cell failures under non-uniform traffic pattern

The proposed algorithm ensures no co-channel interference and achieves high channel utilization. Under our algorithm, a cell (i.e., a mobile base station) may borrow a channel as long as it receives channel usage information from a subset of its neighbors, and there is at least one common available channel in the subset. Therefore, the proposed algorithm is fault-tolerant. Moreover, a cell can lend a channel to multiple neighbors concurrently as long as no two of them are neighbors, and hence increases the probability of a channel being reused.

Chapter 4

Comparison of Two Channel Allocation Approaches: Channel Pre-allocation Vs. Non-Pre-allocation

4.1 Introduction

As mentioned earlier in Chapter 1, channels allocated to a cellular network are limited, and should be reused efficiently. Reusing channels efficiently means that the channel reuse pattern should be compact, that is, a channel used by a cell C_i should be reused by all cells C_j, where the distance between C_i and C_j is equal to or slightly greater than minimum channel reuse distance, i.e., D_{min}.

Channels can be either pre-allocated to cells initially, or can be allocated to cells whenever the need for additional channels arises. We observe that if channels are not pre-allocated to cells, the channel reuse pattern may be non-compact depending on the call arrival pattern. A non-compact channel reuse pattern implies that channels are not reused efficiently.

The disadvantages of the no-channel-pre-allocation scheme can be illustrated in Figure 4.1. In

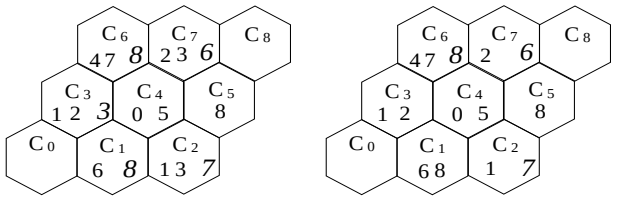


Figure 4.1: Disadvantages of no-channel-pre-allocation

Figure 4.1-(a), there are 9 channels total in the system, with channel ids from 0 to 8. In each cell, channel ids shown in *italics* are the channels available in that cell. For example, channels 6 and 8 are allocated to cell C_1. Channel 6 is being used, while channel 8 is available in C_1. Suppose that

C_4 needs to borrow a channel and it gets channel usage information from all its neighbors. The information is shown in Figure 4.1-(a). Based on this information, C_4 cannot borrow any channel. Channel 7 is available in C_2, however, it is being used by C_6. Cell C_7 has channel 6 available, but channel 6 is being used by C_1. The fact is that although some neighbors have available channels, they do not have common available channels.

Another disadvantage of not pre-allocating channels to cells is illustrated in Figure 4.1-(b). The difference between Figure 4.1-(a) and Figure 4.1-(b) is that all the channels, 0 to 8, have been allocated to cell C_4 and its neighbors in Figure 4.1-(a), while in Figure 4.1-(b), channel 3 is not allocated to C_4 or any of its neighbors. Suppose that C_4 needs to borrow a channel and it gets channel usage information from all its neighbors except C_5 (the channel usage information is shown in Figure 4.1-(b); channel ids in *italics* indicate the channels that are not being used by that cell). Based on this information, C_4 cannot borrow a channel, even though channel 3 is available, since it has not been allocated to C_4 or any of its neighbors yet. This results in lower efficiency. One way to solve these problems appears to be to pre-allocate all of the channels to cells.

In this chapter, we study how channel pre-allocation affects the performance of the channel allocation algorithm. In order to do this, we modify the channel allocation algorithm proposed in Chapter 2 to allow a subset of available channels to be pre-allocated to cells, while the rest are kept in an open pool. By changing the size of the subset of pre-allocated channels, we can pre-allocate to cells all channels, some channels, or no channel at all. We simulate all these cases with respect to the modified channel allocation algorithm and compare their performance.

The rest of this chapter is organized as follows. In Section 4.2, system model is described. Related works are reviewed in Section 4.3. In Section 4.4, the proposed algorithm is given in detail, followed by the correctness proof. In Section 4.5, the performance of the proposed algorithm is evaluated by simulation. Section 4.6 concludes the chapter.

4.2 System Model

We assume a **3-cell cluster model** [33] (described in page 17). We use the terms "cell" and "the base station (BS) in the cell" interchangeably. Figure 4.2 illustrates an example of a cellular network. Each cell has 6 neighbors (by wrapping around the cells at the edge).

In our model, both BSs and communication links could fail. If a BS fails, then all the calls supported by it fail at the same time. A mobile host could fail as well. The failure of a mobile host only affects its ongoing communication.

4.3 Related Works

In [13], the authors propose a channel allocation algorithm for cellular networks. The Update approach (described in page 6) is adopted and all channels are pre-allocated to cells. Channels pre-allocated to a cell are called primary channels of that cell. Primary channels have higher priority to be allocated for calls in each cell. A cell needs to borrow a channel only after it uses up all of its primary channels and a new call originates. Whenever a cell acquires or releases a channel, it informs all of its interference neighbors about this. The proposed algorithm is fault-tolerant. In order to borrow a channel, a cell does not need to receive channel usage information from all its interference neighbors. In [6, 40], the Search approach (described in page 7) is used, instead of the Update approach. Similar to [13], all channels are pre-allocated to cells in [6, 40], and the algorithms proposed there are also fault-tolerant.

In [33], the authors propose a distributed dynamic channel allocation algorithm under a 3-cell cluster model. The Search approach is adopted and channels are not pre-allocated to cells. If a channel is available when a cell C_i needs to support a call, then C_i picks it to use. Otherwise, C_i sends a request message to each of its neighbors, asking for their channel usage information. Based on the information received from all of its neighbors, C_i tries to select a channel r to borrow. C_i can use r only if its request for borrowing r is granted by all of those neighbors to which r has been allocated. After a cell grants a neighbor's request, say C_i's request, for some channel r, it marks r for transfer. After that, it will not grant any other neighbor's request for the same

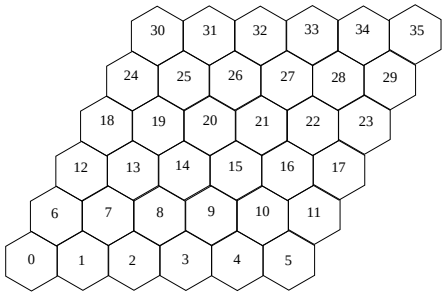


Figure 4.2: A cellular network

channel r even if that neighbor is not a neighbor of C_i; this limits channel reuse. Moreover, the algorithm is not fault tolerant because a cell needs to get channel usage information from all its neighbors to borrow a channel. In [43], the authors propose a channel allocation algorithm under a 3-cell cluster model too. In the algorithm, the Search approach is adopted and channels are not pre-allocated to cells. However, unlike the algorithm proposed in [33], the algorithm in [43] is fault-tolerant in the sense that to borrow a channel, a cell does not need to get channel usage information from all its interference neighbors.

4.4 Adaptive Channel Allocation Algorithm

4.4.1 Basic Idea

The adaptive channel allocation algorithm in this chapter is an extension of the algorithm proposed in Chapter 2. First we review the data structures used in the algorithm. Each cell C_i has 6 neighbors and each neighbor is given a unique neighbor id: *nb_1, nb_2, . . . , nb_6*. The set of neighbors of C_i is denoted by NB_i, i.e., $NB_i := \{nb\_1, nb\_2, nb\_3, nb\_4, nb\_5, nb\_6\}$ (see Figure 4.3).

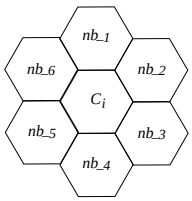


Figure 4.3: Neighborhood relationship in the cellular network

The set of neighbors of C_i are divided into 5 groups:

1. $Group\_1 := \{nb\_1, nb\_4\}$,
2. $Group\_2 := \{nb\_2, nb\_5\}$,
3. $Group\_3 := \{nb\_3, nb\_6\}$,
4. $Group\_4 := \{nb\_1, nb\_3, nb\_5\}$,
5. $Group\_5 := \{nb\_2, nb\_4, nb\_6\}$.

There are N channels total and they are uniquely ordered. Channels are divided into two sets: S_{PRE} and S_{NOPRE}. Channels 0 to M ($0 \leq M < N$) belong to S_{PRE} and channels $M+1$ to $N-1$

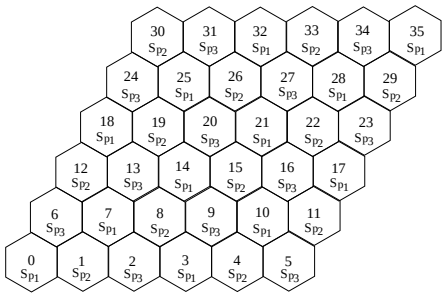


Figure 4.4: Channel pre-allocation pattern

are in S_{NOPRE}. Channels in S_{NOPRE} are not pre-allocated to any cell, while channels in S_{PRE} are pre-allocated to cells according to the following rule:

> **Channel Pre-allocation Rule**: *a channel can be allocated to at most one cell in a cluster of 3 mutually adjacent cells.*

S_{PRE} is divided into three subsets: S_{P1}, S_{P2}, and S_{P3}. Channels in these three subsets are allocated to cells as shown in Figure 4.4. For any cell C_i, S_{C_i} denotes the set of channels that have been pre-allocated to C_i.

When a cell C_i needs a channel, it picks one of the available channels allocated to it with the highest order for use. If no channel allocated to it is available, then it sends a request message to get channel usage information from each neighbor and sets a timer. In this case, it is in Search Mode and it is a borrower. If C_i receives channel usage information from all its neighbors before the timer expires, then it first computes the set of channels that have not been allocated to any of its neighbors as well as itself. It picks such a channel to use if there is one. If there is no such channel, it computes the set of channels allocated to its neighbors that are not being used by any of them. If this set is empty, then it drops the call; otherwise, it selects a channel from this set to borrow. Even if C_i does not receive channel usage information from all its neighbors when the timer expires, it may still be able to borrow a channel. It can do so as long as it receives channel usage information from all members of any group (i.e., *Group*_1 through *Group*_5) and there is a common channel allocated to all members of that group which is not being used by any of these members. For example, in Figure 4.3, suppose cell C_i needs to borrow a channel and it receives channel usage information from *nb_1* and *nb_4* only. Note that *nb_1* and *nb_4* form a group, namely, *Group*_1. If both *nb_1* and *nb_4* have channel r available, then C_i can borrow channel

r. Since channel r is allocated to *nb_1*, it could not have been allocated to *nb_2* or *nb_6* because *nb_2* or *nb_6* is a neighbor of *nb_1*. Similarly, channel r could not have been allocated to *nb_3* or *nb_5* because they are neighbors of *nb_4*. Therefore, if *nb_1* and *nb_4* grant cell C_i's request for borrowing channel r, then C_i can use r without causing co-channel interference.

When a call using a borrowed channel terminates in a cell C_i, C_i checks whether the borrowed channel is in S_{NOPRE}. If yes, then C_i keeps the channel for future use. If the borrowed channel belongs to S_{PRE} (i.e., a channel that was pre-allocated to one of its neighbors), then C_i returns this channel to those neighbors from which it borrowed this channel.

Under our algorithm, a cell is allowed to lend a channel to several borrowers concurrently, as long as no two of them are neighbors. This increases channel reuse. If there are two neighboring cells that are in Search Mode concurrently (i.e., two neighboring borrowers) and each of them borrows a channel, then the algorithm ensures that they do not borrow the same channel (this is proved in Section 4.4.4).

Next, we present the algorithm. The following data structures are used in the algorithm. In the following, we use the terms "cell C_i" and "the base station BS_i in cell C_i" interchangeably.

4.4.2 Data Structures

The data structures maintained by each cell C_i are given in Table 4.1.

Table 4.1: Data Structures

$Spectrum$:	the set of all channels in the system.
S_{PRE}:	the set of channels pre-allocated to cells.
S_{NOPRE}:	the set of channels not pre-allocated to cells.
S_{C_i}:	the set of channels pre-allocated to C_i.
NB_i:	the set of neighbors of C_i.
$Allocate_i$:	the set of channels currently allocated to C_i.
$Busy_i$:	the set of channels currently being used by C_i.
$Transfer_i$:	the set of channels marked for transfer by C_i.
$Granted_i$:	a set of sets in C_i.
$Lent_i$:	a set of sets in C_i.
Num_Reply_i:	number of *replies* C_i gets after the timer expires.

Next, we explain the purpose of the data structures. $|Spectrum| := N$. $S_{PRE} := \{0, 1, \ldots, M\}$, where $0 \leq M < N$. $S_{NOPRE} := \{M{+}1, M{+}2, \ldots, N{-}1\}$. $NB_i := \{nb\_1, nb\_2, nb\_3, nb\_4, nb\_5, nb\_6\}$.

Initially, $Allocate_i$ is equal to S_{C_i}. At any given time, $Allocate_i \supseteq S_{C_i}$. $Transfer_i$ is the set of channels that C_i plans to transfer. At any given time, $Transfer_i \cap Busy_i$ is an empty set, and $Transfer_i \subseteq Allocate_i$. In cell C_i, $Granted_i$ is a set of sets. $\forall\ r \in Allocate_i$, $Granted_i(r)$ denotes the set of cells to which C_i has sent an *agree(r)* message. An *agree(r)* message from C_i to C_j means that C_i agrees to grant C_j's request to borrow channel r. C_i lends channel r to C_j only when it is notified by C_j to release the channel. $Lent_i$ is also a set of sets maintained by cell C_i. $\forall\ r \in Allocate_i$, $Lent_i(r)$ denotes the set of cells to which C_i has lent channel r. Initially, $Busy_i$, $Transfer_i$, $Granted_i(r)$ and $Lent_i(r)$ are all empty sets. Each message is assigned a timestamp, using Lamport's timestamp [23], the lower the timestamp, the higher the priority. Obsolete messages are ignored by comparing timestamps.

4.4.3 The Algorithm

Formal description of the proposed algorithm is given in Table 4.2.

4.4.4 Correctness of the Algorithm

In this subsection, we prove that the algorithm is correct and it is deadlock-free.

Lemma 4.4.1 *Two neighboring cells, C_i and C_j, are not allowed to borrow the same channel concurrently under our algorithm.*

Proof: Let r_1 be the channel borrowed by C_i, r_2 be the channel borrowed by C_j. Without loss of generality, we assume that C_i's request has lower priority than C_j's request. We prove that $r_1 \neq r_2$. Since C_j's request has higher priority, C_j will defer sending a reply to C_i until it finishes its channel borrowing process (note that we use Lamport's timestamp to determine the priority of the request message). When C_j borrows channel r_2 to use, it adds r_2 to $Busy_j$. If $r_2 \in S_{NOPRE}$, it also adds r_2 to $Allocate_j$. Then, it sends a reply message to C_i. The following two cases arise for cell C_i.

- case 1: C_i borrows r_1 as a result of getting replies from all of its neighbors for its request. Following two sub-cases arise.
  - (1): $r_2 \in S_{NOPRE}$. If $r_1 \in Free_i := Spectrum - \cup Allocate_k$ (step C-1-α in the algorithm), then, $r_1 \neq r_2$, since $(r_2 \in Allocate_j) \wedge (C_j \in NB_i)$. If $r_1 \in Free_i :=$

Table 4.2: Our Adaptive Channel Allocation Algorithm

A: When a cell C_i needs a channel to support a call, $Free_i := Allocate_i - Busy_i - Transfer_i$.
If $Free_i = \emptyset$, it sets a timer and sends a *request* message to each neighbor;
else, it picks $r \in Free_i$ with the highest order to support the call, $Busy_i := Busy_i \cup \{r\}$.

B: When a cell C_j receives a *request* from C_i, if its *request* has a lower timestamp than C_i's *request*,
it defers sending a *reply* to C_i; otherwise, it sends to C_i a *reply* message, including $Allocate_j$,
$Busy_j$ and $Interfere_j$. $Interfere_j$ is computed as follows: (1): $Temp_j := \emptyset$,
(2): $\forall r \in Transfer_j$, if $(Granted_j(r) \cup Lent_j(r)) \cap (NB_i \cup \{C_i\}) \neq \emptyset$, $Temp_j := Temp_j \cup \{r\}$, and
(3): $Interfere_j := Busy_j \cup Temp_j$.

C: After C_i receives a *reply* message from each neighbor or after its timer expires, $Free_i := \emptyset$.
1: If $Num\_Reply_i = 6$,
α: If $(Free_i := Spectrum - \cup Allocate_k) \neq \emptyset$, $k \in \{nb\_1, nb\_2, nb\_3, nb\_4, nb\_5, nb\_6, i\}$,
it picks $r \in Free_i$ with the highest order to support the call. $Allocate_i := Allocate_i \cup \{r\}$,
$Busy_i := Busy_i \cup \{r\}$. $Granted_i(r) := \emptyset$, and $Lent_i(r) := \emptyset$.
β: Otherwise, $Free_i := Spectrum - \cup Busy_k$. If $Free_i = \emptyset$, it drops the call; else it sets
a new timer, picks a channel $r \in Free_i$ with the lowest order, and sends *transfer(r)* messages to
neighbors to which r has been allocated.
2: If $Num\_Reply_i < 6$:
α: If $Num\_Reply_i = 2$, let x and y be the two neighbors from which a *reply* has been received.
If $((\exists m \in \{1,2,3\}) \wedge (\{x,y\} = Group\_m))$, $Free_i := \cap_{k \in Group\_m}(Allocate_k - Interfere_k)$.
β: If $Num\_Reply_i = 3$, let x, y and z be the three neighbors from which a *reply* has been received.
If $((\exists m \in \{1,2,3\}) \wedge (\{x,y,z\} \supset Group\_m))$, $Free_i := \cap_{k \in Group\_m}(Allocate_k - Interfere_k)$;
else if $((\exists m \in \{4,5\}) \wedge (\{x,y,z\} = Group\_m))$, $Free_i := \cap_{k \in Group\_m}(Allocate_k - Interfere_k)$.
γ: If $Num\_Reply_i = 4$, let x and y be the two neighbors from which a *reply* has **NOT** been received.
If $\exists m \in \{1,2,3\} \wedge (NB_i - \{x,y\}) \supset Group\_m$, $Free_{i\_m} := \cap_{k \in Group\_m}(Allocate_k - Interfere_k)$.
If $\exists n \in \{4,5\} \wedge (NB_i - \{x,y\}) \supset Group\_n$, $Free_{i\_n} := \cap_{k \in Group\_n}(Allocate_k - Interfere_k)$.
$Free_i := Free_{i\_m} \cup Free_{i\_n}$.
δ: If $Num\_Reply_i = 5$, let x be the neighbor from which a *reply* message has **NOT** been received.
If $\exists m \in \{1,2,3\} \wedge (NB_i - \{x\}) \supset Group\_m$, $Free_{i\_m} := \cap_{k \in Group\_m}(Allocate_k - Interfere_k)$.
If $\exists n \in \{4,5\} \wedge (NB_i - \{x\}) \supset Group\_n$, $Free_{i\_n} := \cap_{k \in Group\_n}(Allocate_k - Interfere_k)$.
$Free_i := Free_{i\_m} \cup Free_{i\_n}$.
If $Free_i = \emptyset$, it drops the call. Otherwise, if $(Temp := Free_i \cap S_{NOPRE}) \neq \emptyset$, it picks a channel
$r \in Temp$ with the lowest order. If $Temp = \emptyset$, it picks a channel $r \in Free_i$ with the lowest order.
C_i sets a new timer and sends a *transfer(r)* message to neighbors to which r has been allocated.

D: When C_j receives a *transfer(r)* message from C_i:
If $((r \notin Busy_j) \wedge (\forall\, C_m \in Granted_j(r) \cup Lent_j(r),\, C_m \notin NB_i))$, it sends an *agree(r)* to C_i,
adds C_i to $Granted_j(r)$, and if $((r \in Allocate_j) \wedge (r \notin Transfer_j))$, it adds r to $Transfer_j$.
Otherwise, it sends a *refuse(r)* message to C_i.

E: After C_i gets a response corresponding to each of its *transfer(r)* before the new timer expires,
If all the responses are *agree(r)* messages, it uses r to support the call, sets $Busy_i := Busy_i \cup \{r\}$
and sends a *release(r)* message to the neighbors from which an *agree(r)* was received.
If $r \in S_{NOPRE}$, $Allocate_i := Allocate_i \cup \{r\}$, $Granted_i(r) := \emptyset$, and $Lent_i(r) := \emptyset$.
Otherwise, it sends a *keep(r)* message to neighbors from which it received an *agree(r)* message.
$Free_i := Free_i - \{r\}$, and it tries to borrow another channel if $Free_i \neq \emptyset$.

F: If the new timer expires before C_i gets a response corresponding to each of its *transfer(r)* message,
it drops the call and sends a *keep(r)* message to each neighbor from which it received an *agree(r)*.

G: After C_i supports or drops a call, it sends a *reply* to neighbors whose *request* has been deferred.

H: When a call using a channel r terminates in cell C_i,
$Busy_i := Busy_i - \{r\}$. If $((r \in S_{PRE}) \wedge (r \notin S_{C_i}))$, C_i sends a *return(r)* message to each neighbor
C_j such that $r \in S_{C_j}$.

I: When C_j receives a *release(r)* message from C_i,
$Granted_j(r) := Granted_j(r) - \{C_i\}$ and $Lent_j(r) := Lent_j(r) \cup \{C_i\}$.
If $(r \in Allocate_j \cap S_{NOPRE})$, $Allocate_j := Allocate_j - \{r\}$, $Transfer_j := Transfer_j - \{r\}$.

J: When a cell C_j receives a *keep(r)* message from cell C_i, $Granted_j(r) := Granted_j(r) - \{C_i\}$;
if $((r \in Allocate_j \cap S_{NOPRE}) \wedge (Granted_j(r) = \emptyset))$, then it sets $Transfer_j := Transfer_j - \{r\}$;
if $((r \in S_{C_j}) \wedge (Granted_j(r) \cup Lent_j(r) = \emptyset))$, then it sets $Transfer_j := Transfer_j - \{r\}$.

K: When C_j gets a *return(r)* from C_i: it sets $Lent_j(r) := Lent_j(r) - \{C_i\}$;
if $Granted_j(r) \cup Lent_j(r) = \emptyset$, it sets $Transfer_j := Transfer_j - \{r\}$.

$Spectrum - \cup Busy_k$ (step C-1-β in the algorithm), then $r_1 \neq r_2$, because $(r_2 \in Busy_j) \wedge (C_j \in NB_i)$.

- (2): $r_2 \in S_{PRE}$. If $r_1 \in Free_i := Spectrum - \cup Allocate_k$ (step C-1-α in the algorithm), then, $r_1 \neq r_2$, because $r_1 \in S_{NOPRE}$ (note that $\cup Allocate_k \supseteq S_{PRE}$ and $Free_i \subseteq S_{NOPRE}$). If $r_1 \in Free_i := Spectrum - \cup Busy_k$ (step C-1-β in the algorithm), then $r_1 \neq r_2$, because $r_2 \in Busy_j \wedge C_j \in NB_i$.

- case 2: C_i borrows r_1 as a result of getting replies from only a subset of its neighbors for its request. In this case, C_i must have borrowed r_1 from some $Group\_m$ ($m \in \{1, \ldots, 5\}$). $r_1 \in (Free_i := \cap_{k \in Group\_m}(Allocate_k - Interfere_k))$ (step C-2 in the algorithm). Because $Interfere_k \supseteq Busy_k$, we have $r_1 \notin Busy_k$. Following two sub-cases arise.

  - $C_j \in Group\_m$. Because C_j sends a reply message only after it finishes its channel borrowing process, $r_2 \in Busy_j$. Thus, $r_1 \neq r_2$ since $r_1 \notin Busy_j$.

  - $C_j \notin Group\_m$. Because C_i and C_j are neighbors, they have two common neighbors. Let them be C_m and C_n. $Group\_m$ contains at least one of C_m and C_n. Thus, we have $r_1 \in (Allocate_{C_m} \cup Allocate_{C_n})$. Suppose $r_1 = r_2 = r$. There are two sub-cases:

    * (1): $r \in S_{PRE}$. We have $r \in S_{C_m}$ (note that $S_{C_m} = S_{C_n}$). It follows that C_m and C_n grant both C_i's and C_j's request to borrow channel r. However, this is impossible. According to our algorithm, a cell does not grant two cells' requests for the same channel if the two cells are neighbors. This is done in Step D of the algorithm. Thus, $r_1 \neq r_2$.

    * (2): $r \in S_{NOPRE}$. There are three sub-cases:

      · (A): $r \in Allocate_{C_m} - Allocate_{C_n}$ just before C_i and C_j borrow channel r. It follows that C_m grants both C_i's and C_j's request for channel r. This is impossible due to the same reason mentioned above.

      · (B): $r \in Allocate_{C_n} - Allocate_{C_m}$. This is similar to sub-case (A).

      · (C): $r \in Allocate_{C_m} \cap Allocate_{C_n}$. It follows that C_m and C_n grant both C_i's and C_j's request for channel r. This is similar to case 2-(1).

    Thus, $r_1 \neq r_2$.

Lemma 4.4.2 *Two neighboring cells are not allowed to use the same channel concurrently under our channel allocation algorithm.*

Proof: Channels in set S_{PRE} are allocated to cells in such a way that the same channel is not allocated to neighboring cells. Moreover, neighboring cells do not borrow the same channel concurrently. It follows that neighboring cells do not use the same channel concurrently. □

Lemma 4.4.3 *The proposed channel allocation algorithm is deadlock-free.*

Proof: In our algorithm, a timeout mechanism is used for getting response for request messages as well as transfer messages. So, hold and wait situation does not arise. Therefore, the algorithm is deadlock-free. □

4.5 Performance Evaluation

4.5.1 Definitions

In this section, we study the performance of our algorithm by varying the size of pre-allocated channels and show how channel pre-allocation improves the performance of the algorithm. We evaluated the performance with respect to three parameters, namely, *call blocking rate*, *handoff drop rate*, and *call failure rate*. Call blocking rate is defined as the ratio of the number of new calls dropped (i.e., blocked new calls) to the total number of new calls. Handoff drop rate is defined as the ratio of the number of inter-handoff calls dropped to the total number of inter-handoff calls. Call failure rate is defined as the ratio of total number of calls dropped (including blocked new calls and dropped inter-handoff calls) to the total number of calls processed. Call arrival rate is defined as the number of call requests per cell per hour.

4.5.2 Simulation Parameters

To evaluate the performance of the algorithm, we used CSIM18 [1] Simulation Engine, which is a process-oriented, discrete-event simulator. The simulated cellular network consists of $6 * 6$ cells. Each cell has 6 neighbors (by wrapping around the cells at the edge). There are a total of 300

[1]CSIM18 Simulation Engine is a product of Mesquite Software, Inc.

channels available in the system. The number of channels in S_{PRE} is a simulation parameter. We investigated the effect of this parameter on the performance of the proposed algorithm.

We assume that the average one-way communication delay between two cells is 4 milliseconds. This average delay includes transmission delay, propagation delay and the message processing time. In the simulation, once a mobile host is generated, it sends a call request to the base station in the cell. Upon receiving such a request, the base station tries to allocate a channel to support the call by using the proposed channel allocation algorithm. If no channel can be found to support the call, then the call is dropped and is counted as a call failure. If a channel can be allocated to support the call, then the mobile host will use this channel for its communication.

We assume that the average service time per call is 3 minutes. During communication, the mobile host may move from one cell to an adjacent cell (i.e., an inter-handoff occurs). If this happens, it releases the channel that is supporting the call to the cell from which it is moving and sends a call request to the cell to which it is moving. The new cell to which it is moving is responsible to allocate a new channel to support the inter-handoff call. If no channel can be allocated for this inter-handoff call, then it is dropped and counted as an inter-handoff failure. At the end of the simulation, the number of inter-handoff failures, the number of call failures and the total number of processed calls were collected.

The simulation was conducted under a non-uniform traffic pattern, which is more realistic. Under non-uniform pattern, a cell can be in normal state or hot state. When a cell is in normal state, call arrival rate is low, and inter-handoff rate is high. When a cell is in hot state, call arrival rate is high and inter-handoff rate is low. The parameters for the non-uniform traffic pattern are given in Table 4.3 [6].

Table 4.3: Parameters for non-uniform traffic pattern

Mean call arrival rate in a normal cell	λ
Mean call 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 changing from normal to hot state	$1/1800s$
Mean rate changing from hot to normal state	$1/180s$
Mean service time per call	$180s$

4.5.3 Simulation Results

We conducted our simulation under 144 different scenarios (6 different call arrival rates, 4 different values of the size of set S_{PRE}, and 6 different number of cell failures, $6 \times 4 \times 6 = 144$). Simulation was run ten times under each scenario, each time with a different seed. Thus, our simulation run total 1440 times. In each run of the simulation, $40,000$ to $60,000$ mobile hosts were generated. Once generated, each mobile host generates one or more calls, including new calls and inter-handoff calls. In order to remove the start-up transients, data was collected only after the first $10,000$ calls were processed. The simulation ended after $100,000$ calls were processed. In the simulation, there are 300 channels in total. The size of the set S_{PRE} is set to be 0, 150, 210 and 300. By changing the size of S_{PRE}, we investigated the effect of the size of pre-allocated channels on the performance of the proposed adaptive algorithm. The simulation was conducted under two scenarios: without cell failures and with cell failures.

1. **Without Cell Failures:** In this set of experiments, no cell fails. The simulation results are shown in Figure 4.5. As seen from these figures, all the three metrics increase as call arrival rate increases. Moreover, the more channels pre-allocated to cells, the better the performance.

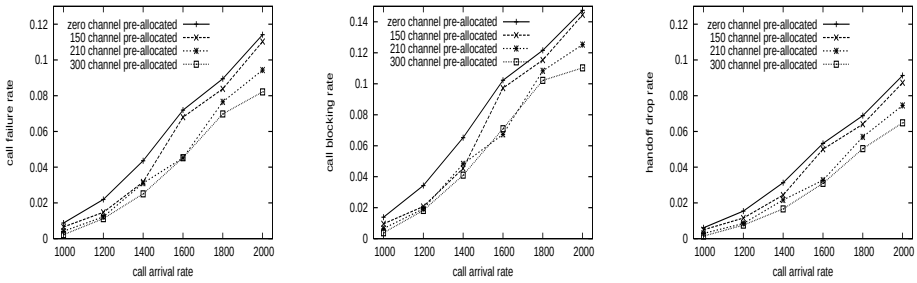


Figure 4.5: Performance without cell failure under non-uniform traffic pattern

2. **With Cell Failures:** We evaluated the performance of our algorithm with cell failures. We set the size of the set S_{PRE} to be 0, 150, 210, and 300 as before, and varied the number of cell failures from 1 to 5. The simulation was run under exactly the same scenario, except that each time the number of cell failures is different. The simulation results are shown in Figure 4.6 to Figure 4.10.

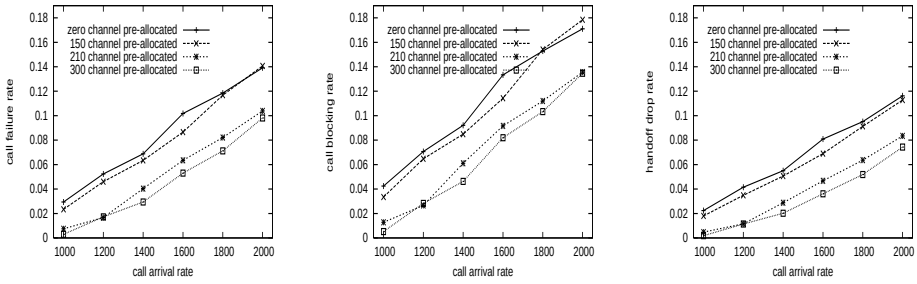


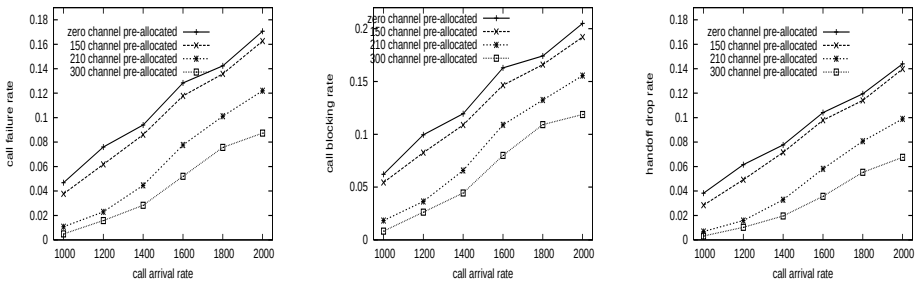
Figure 4.6: Performance with one cell failure under non-uniform traffic pattern

We can see from these figures that all the three metrics increase as call arrival rate increases. Moreover, the three metrics also increase with the number of cell failures. In all the cases (from one cell failure to five cell failures), the one with 300 pre-allocated channels always outperforms the other three (i.e., the ones with 0, 150, and 210 pre-allocated channels), in terms of all the metrics. This better performance is attributed to the channel pre-allocation feature of our algorithm: the more pre-allocated channels, the more likely to find an available channel to use (or borrow), causing a lower call failure rate, call blocking rate, and handoff drop rate.

From the figures, we can see that the proposed algorithm can tolerate cell failures very well. In Figure 4.10-(a), under non-uniform traffic pattern and with five cell failures, call failure rate with 300 pre-allocated channels is 9.11% when call arrival rate is 2000. In other words, when 13.89% of the cells (there are 36 cells total) in the system fail and when the system is heavily loaded (represented by a high call arrival rate), 90% of the calls can still be supported. In Figure 4.5-(a), under the same traffic pattern and the same call arrival rate, but without cell failure, call failure rate with 300 pre-allocated channels is 8.21%. Thus, as cell failure rate increases from 0% to 13.89%, we only observe an increase in call failure rate of about 1%.

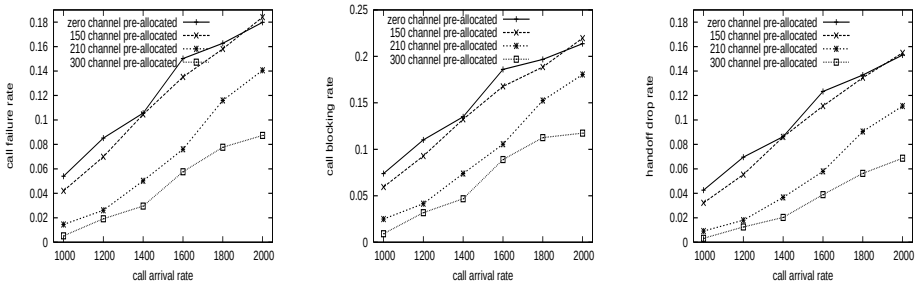
4.6 Conclusion

To our knowledge, the effect of pre-allocating channels to cells on the performance of channel allocation has not been studied quantitatively. In this chapter, we presented an adaptive distributed channel allocation algorithm which allows the flexibility of pre-allocating different number



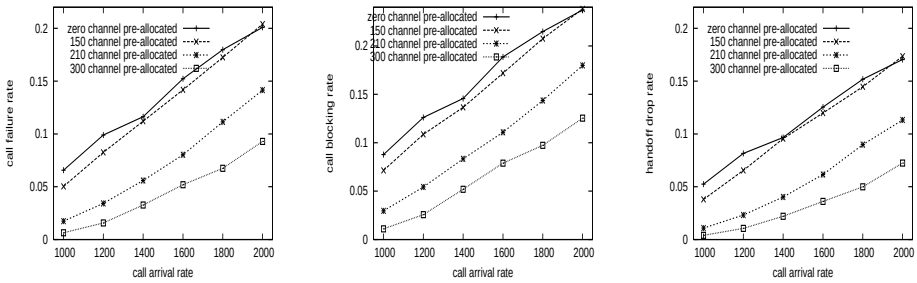
(a) (b) (c)

Figure 4.7: Performance with two cell failures under non-uniform traffic pattern



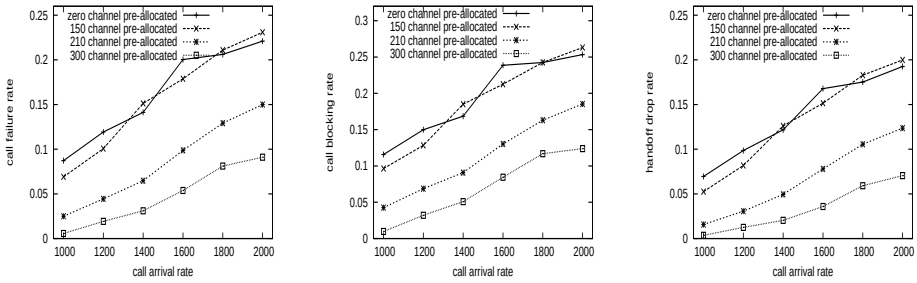
(a) (b) (c)

Figure 4.8: Performance with three cell failures under non-uniform traffic pattern



(a) (b) (c)

Figure 4.9: Performance with four cell failures under non-uniform traffic pattern



(a) (b) (c)

Figure 4.10: Performance with five cell failures under non-uniform traffic pattern

of channels to cells. The simulation results show that channel pre-allocation helps in lowering call blocking rate, handoff drop rate, and call failure rate. More the channels pre-allocated, better the performance. Therefore, we conclude that distribute channel allocation algorithms based on total pre-allocation of channels will have better performance than algorithms that do not pre-allocate channels.

Chapter 5

A Distributed Fault-Tolerant Channel Allocation Algorithm for Cellular Networks Under Resource Planning Model

5.1 Introduction

From the results in Chapter 4, it is clear that a channel allocation algorithm benefits from pre-allocating all channels to cells, which is an example of Resource Planning Model (described on page 7). In this chapter, we present a more general algorithm for channel allocation under Resource Planning Model.

A channel allocation algorithm usually has two parts: a *channel acquisition algorithm* and a *channel selection algorithm*. The task of the former is to compute the set of channels that are not being used by cells within distance D_{min}. The goal of the latter is to choose a channel from the computed set of channels smartly so that good channel reuse pattern can be achieved. Channel selection algorithm is very important because it affects channel utilization. A good channel selection algorithm improves channel utilization.

The main contribution of this chapter is that we propose a distributed and fault-tolerant channel allocation algorithm which reuses channels efficiently. It is fault-tolerant because a cell does not need to get channel usage information from all its interference neighbors to borrow a channel. It includes a new channel selection algorithm which takes into account the interference caused by borrowing a channel to the cells which have the channel allocated to them. The selection algorithm chooses a channel in such a way that it increases the chance of reusing the same channel

and hence increasing channel utilization.

The rest of this chapter is organized as follows. System model is given in Section 5.2. Related works are reviewed in Section 5.3. Section 5.4 presents the details of the distributed channel allocation algorithm. The performance analysis of the proposed algorithm and the simulation results are given in Section 5.5 and Section 5.6. Section 5.7 concludes the chapter.

5.2 System Model

We assume the Resource Planning Model, under which the set of all cells is divided into k disjoint subsets, S_0, S_1, ..., S_{k-1}, such that in the same subset, the distance between any two cells is at least D_{min}. The set of all channels is divided into k disjoint subsets, PC_0, PC_1, ..., PC_{k-1}, correspondingly. Channels in PC_i are allocated to the cells in S_i and are called primary channels of cells in S_i and secondary channels of cells in S_j $(i \neq j)$. Cells in S_i are called primary cells of channels in PC_i and secondary cells of channels in PC_j $(i \neq j)$. Primary channels of a cell, say C_i, have higher priority to be allocated to support a call in C_i than secondary channels. A secondary channel is used to support a call only if there is no primary channel available. This is illustrated in Figure 5.1. The set of all 81 cells is divided evenly into 9 disjoint subsets, namely, A, B, C, D, E, F, G, H and I. Each subset has 9 cells. For example, subset A consists of cells A_0, A_1, A_2, A_3, A_4, A_5, A_6, A_7, and A_8. $D_{min} = 3\sqrt{3}\ R$ where R is the cell radius. The distance between any two cells in the same subset is equal to D_{min}. The set of channels available in the system is divided evenly into 9 disjoint subsets, namely, PC_A, PC_B, PC_C, PC_D, PC_E, PC_F, PC_G, PC_H, and PC_I.

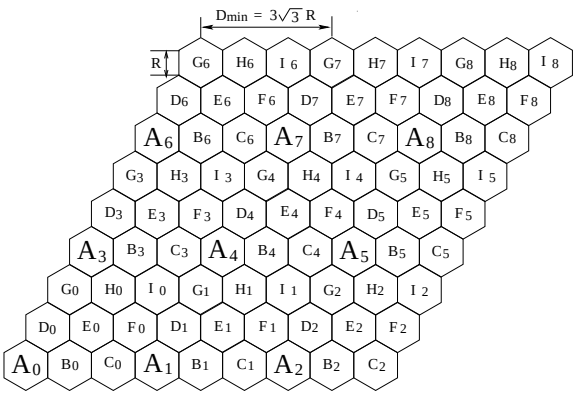


Figure 5.1: A partition of a cellular network

Following are some definitions about Resource Planning.

Definition 5.2.1 *For a given cell C_i, its interference neighbors, denoted by INb_i, is the set of cells, C_j, that are at a distance less than D_{min} from C_i. i.e., $INb_i := \{C_j \mid dist(C_i, C_j) < D_{min}\}$, where $dist(C_i, C_j)$ is the distance between C_i and C_j.*

We assume that cells wrap around. Therefore, each cell in Figure 5.1 has 30 interference neighbors. For example, the interference neighbors of cell A_4 are the cells: B_1, B_3, B_4, B_6, C_1, C_3, C_4, C_6, D_1, D_2, D_3, D_4, E_1, E_3, E_4, F_0, F_1, F_3, F_4, G_1, G_2, G_3, G_4, H_0, H_1, H_3, H_4, I_0, I_1, and I_3, a total of 30 interference neighbors. If a channel r is being used in cell A_4, to avoid co-channel interference, none of its interference neighbors can use r concurrently. But r can be reused in cells which are at a distance greater than or equal to D_{min} from A_4.

Definition 5.2.2 *For a cell $C_i \notin S_j$ and a channel $r \in PC_j$, the set of interference primary cells of channel r relative to C_i, denoted by $IPC_i(r)$, is the intersection of the set of cells that are primary cells of r and the set of cells that are interference neighbors of C_i. i.e., $IPC_i(r) := S_j \cap INb_i$.*

For example, in Figure 5.1, suppose channel r has been allocated to subset B, then B_1, B_3, B_4 and B_6 are in $IPC_{A_4}(r)$. Note that for any cell $C_i \notin S_j$ and a channel $r \in PC_j$, there are at most 4 cells in $IPC_i(r)$. For any cell C_i, all its interference neighbors fall into several subgroups , with neighbors in the same subset falling into the same subgroup, we call it a *complete subgroup* of C_i. The *complete subgroup* of C_i containing neighbors in subset S_n is denoted by $sub_i(S_n)$, where $n \in \{0, 1, \ldots, k-1\}$. Cells in the same *complete subgroup* of C_i have the same set of primary channels. For example, in Figure 5.1, cell A_4's 30 interference neighbors fall into 8 subgroups, with neighbors in the same subset in one subgroup, i.e., there are 8 *complete subgroups* of A_4: $sub_{A_4}(B)$, $sub_{A_4}(C)$, $sub_{A_4}(D)$, $sub_{A_4}(E)$, $sub_{A_4}(F)$, $sub_{A_4}(G)$, $sub_{A_4}(H)$, and $sub_{A_4}(I)$. All the neighbors of cell A_4 that are in the same subset fall into the same complete subgroup. For example, $sub_{A_4}(B)$ includes B_1, B_3, B_4 and B_6, and $sub_{A_4}(H)$ includes H_1, H_3 and H_4. Following are some properties of the Resource Planning Model [6, 13].

Property 5.2.1 $\forall C_x, C_y \in S_i \ (x \neq y),\ dist(C_x, C_y) \geq D_{min}$.

Property 5.2.2 *For any two cells C_x ($C_x \in S_i$) and C_y ($C_y \in S_j$), and $i \neq j$, such that they are in each other's interference neighborhood and are requesting for the same channel r ($r \notin PC_i \wedge r \notin PC_j$), then they have at least one interference neighbor in common, which is a primary cell of r. That is:* $\forall\, C_x \in S_i,\, C_y \in S_j : C_x \in INb_y \implies \forall\, r : r \notin PC_i \wedge r \notin PC_j,\, IPC_x(r) \cap IPC_y(r) \neq \emptyset$.

For example, in Figure 5.1, C_4 and A_4 are in each other's interference neighborhood. Suppose they are requesting for the same channel r which is a primary channel of cells in subset B, then they have at least one common interference neighbor, which is a primary cell of r. B_4 is such a common neighbor in this case. If we take C_3 and A_4 as an example, then B_1, B_3, B_4, and B_6 are the common interference neighbors of both C_3 and A_4.

5.3 Related Works

In [5, 7], the Resource Planning Model is adopted. When a cell needs a channel to support a call, if there are available primary channels allocated to the cell, then the cell selects one such channel to support the call without consulting its neighbors. Otherwise, the cell tries to borrow a secondary channel by sending request messages to its interference neighbors, asking for their channel usage information. The cell can borrow a channel from its neighbors as long as using this channel causes no co-channel interference. To ensure this, this cell consults its neighbors before it uses the borrowed channel. When the call terminates, the borrowed channel is returned to the cell from which it was borrowed. In these algorithms, if a cell wants to borrow a channel, it has to wait until it receives channel usage information from all its interference neighbors. Thus, these algorithms are not fault-tolerant.

In [6], a fault-tolerant channel allocation algorithm is proposed. Like [13, 5, 7], *Resource Planning Model* is adopted in [6]. Unlike [13], the algorithm proposed in [6] adopts a Search approach. We illustrate how the algorithm proposed in [6] works with an example using Figure 5.2.

Figure 5.2 contains cell A_4 and all of its 30 interference neighbors from Figure 5.1. The 30 neighbors of A_4 fall into 8 subsets. In Figure 5.2, we only highlight the neighbors in subset B and neighbor G_1 for the purpose of illustration of the algorithm. Neighbors in the same subset have the same set of primary channels. Suppose cell A_4 needs to borrow a channel from its neighbors.

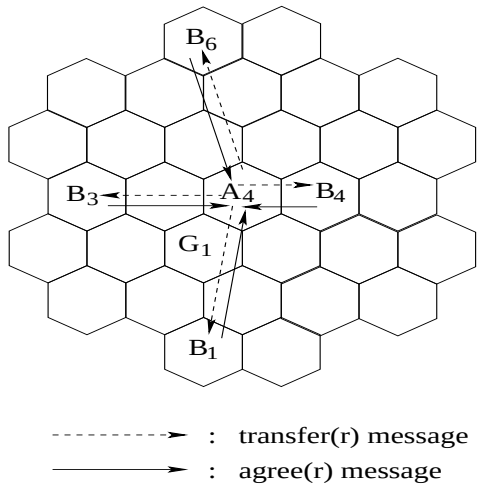


Figure 5.2: Illustration of the algorithm proposed in [6]

It sends request messages to all its neighbors, asking for their channel usage information. Upon receiving a request message from cell A_4, each neighbor of A_4 will send back to A_4 a reply message, including the set of primary channels that are available for lending to A_4. Assuming that cell A_4 only gets reply messages from neighbors B_1, B_3, B_4, and B_6, cell A_4 begins to compute the set of channels that it may borrow. If there is a primary channel r that is not being used by all the cells in the set $\{B_1, B_3, B_4, B_6\}$, then A_4 can borrow the channel r by sending a transfer(r) message to these four neighbors. Cell A_4 borrows channel r successfully if all these four neighbors agree to grant channel r for A_4 to use. After cell A_4 finishes using the borrowed channel r, it returns channel r to these four neighbors. If not all these four neighbors of A_4 agree to grant channel r for A_4 to use, then A_4 cannot borrow r successfully. In such a case, it will notify these four neighbors about its failure to borrow channel r.

For each primary channel r, cell B_1 (B_3, B_4, B_6) maintains a set $I_{B_1}(r)$ ($I_{B_3}(r)$, $I_{B_4}(r)$, $I_{B_6}(r)$ respectively). $I_{B_1}(r)$ records all the neighbors to which B_1 grants channel r. Assuming that $I_{B_1}(r)$ is empty and r is an available channel when B_1 receives A_4's transfer(r) message, then B_1 will grant channel r for A_4 to borrow. B_1 adds A_4 into set $I_{B_1}(r)$. Suppose G_1 also needs to borrow a channel from neighbors and sends request messages to all its interference neighbors, including cell B_1. When B_1 receives the request message from G_1, it computes the set of available primary channels and attaches this set of channels to its reply to G_1. A primary channel r can be included in this set of channels if the following two conditions are satisfied: (1) r is not being used by B_1, and (2) for each cell $C_k \in I_{B_1}(r)$, G_1 and C_k are not interference neighbors. Since G_1 and A_4 are

interference neighbors, and $A_4 \in I_{B_1}(r)$, B_1 will not include channel r in its reply message to cell G_1.

A disadvantage of this algorithm is that the whole pool of available channels is not reused efficiently. Cell B_1 could include channel r in the set of channels attached to its reply to G_1 even though G_1 and A_4 are neighbors, because B_1's permission for A_4 to use channel r does not mean that A_4 has already acquired channel r. To acquire a secondary channel r, A_4 needs to get permission from all its neighbors to which channel r is allocated as a primary channel (i.e., B_1, B_3, B_4, and B_6). Because A_4 may not be able to acquire channel r, G_1 should be given at least a chance to borrow channel r. If A_4 is not successful in borrowing r, then excluding r from the set of channels attached to B_1's reply to G_1 makes the size of the set of available channels smaller than it should be. Thus the available channels are not reused efficiently. In this chapter, we address this issue and propose a better algorithm which makes use of the available pool of channels efficiently.

5.4 A Distributed Channel Allocation Algorithm

5.4.1 Basic Idea

Each message is timestamped with Lamport's timestamp [23] (described on page 8). Outdated messages can be detected by comparing timestamps and discarded. For each primary channel r of a cell C_i, C_i keeps track of the set of cells which borrowed the channel r from C_i and have not released it yet. Let's denote this set as $Lent_i(r)$.

When cell C_i needs a channel to set up a call, it assigns a primary channel to support the call if there exists such a primary channel. Otherwise, it sends request message to all its interference neighbors, asking for their channel usage information. When such a request message is received, each cell C_j $(j \neq i)$ will check whether a certain primary channel r can be included in its reply message. Cell C_j includes a primary channel r in its reply message to cell C_i if C_j is not currently using channel r and $(C_i \cup INb_i) \cap Lent_j(r)$ is an empty set. The basic idea behind this is that as long as C_j is not using a primary channel r, and none of the cells to which it lent r is an interference neighbor of C_i, C_j can lend r to C_i. For example, suppose r is a primary channel of cell C_4 in Figure 5.1 and $Lent_{C_4}(r)$ is an empty set. Assuming that C_4 first lends the channel r to D_5, thus, $Lent_{C_4}(r) := \{D_5\}$. Also, suppose C_4 granted cell H_1's request to borrow channel r, but

has not received any response from H_1 yet (note that H_1 is not included in set $Lent_{C_4}(r)$). Then suppose C_4 receives a request message from A_4. Because C_4 is not using r, and cell D_5 which is using r is not in A_4's interference neighborhood, C_4 can include channel r in its reply to A_4.

In our model (shown in Figure 5.1), a cell C_i has 30 interference neighbors. These neighbors fall into 8 *complete subgroups* with respect to C_i. In order to borrow a channel, C_i does not have to wait until it receives channel usage information from all its interference neighbors. For example, in Figure 5.2, suppose cell A_4 needs to borrow a channel and it only receives channel usage information from neighbor B_1, B_3, B_4 and B_6. These four neighbors form a *complete subgroup* with respect to cell A_4. If there is a common available channel in this *complete subgroup* [1], then A_4 can borrow it.

If two interference neighbors, C_i and C_j, want to borrow the same secondary channel r at the same time, then based on Property 5.2.2 of *Resource Planning*, they have at least one common interference neighbor which is a primary cell of channel r. Let C_k be such a cell. At most one borrower, C_i or C_j, can get a *grant*[2] message from C_k.

5.4.2 The Channel Selection Algorithm

Given a set of channels available to borrow, how a cell chooses a channel to borrow is called the channel selection problem. We adopt a priority based channel selection algorithm that assigns a priority to each channel to borrow. A cell always selects the channel with the highest priority to borrow.

Next, we explain how to compute the priority for each channel. For each primary channel r of cell C_j, C_j keeps track of the set of cells which borrowed r successfully from it and have not released r yet. When C_j receives a request message from C_i (i.e., C_j and C_i are interference neighbors), it computes the set of primary channels available and includes them in the reply message to C_i. For each primary channel r of cell C_j, if C_j is using r, or it has lent r to a neighbor C_k such that C_i and C_k are neighbors, then r will not be included in the reply message to C_i, thus, C_i will not be able to borrow r. Otherwise, C_j assigns r a priority, denoted by $p_r(j)$, and includes r together with its priority in the reply message to C_i. **The rules to assign priority to each primary**

[1] A common available channel of cells in a *complete subgroup* is a common primary channel of these cells which is not being used by any of them.

[2] *grant* message means that the cell agrees that the borrower can use the channel.

channel r included in C_j's reply to neighbor C_i are as follows:

- (1): if C_j has lent r to some neighbors, and none of them is a neighbor of C_i, then it assigns a high priority H to r, i.e., $p_r(j) = H$;
- (2): if C_j has sent a *grant(r)* message in response to some of its neighbors for the same primary channel r, and at least one such neighbor is a neighbor of C_i, then it assigns a low priority L to r, i.e., $p_r(j) = L$. Otherwise, (i.e., for each cell C_k to which C_j has sent *grant(r)* message, C_k is not an interference neighbor of C_i) $p_r(j) = H$ (note that if the priority of r has already been assigned in step (1), it will be re-assigned in this step);
- (3): if primary channel r is an available channel in C_j (i.e., C_j neither lent nor granted r to any of its neighbors), then C_j assigns medium priority M to r, i.e., $p_r(j) = M$ (where $H \gg M > L$).

After assigning priority, C_j includes the information about the available primary channels with their priorities in the reply message and sends the reply message to C_i. When C_i receives the channel usage information included in reply messages from neighbors, it computes the set of channels that it can borrow. For each secondary channel r in this computed set, C_i computes the priority of r. Let us assume that there are totally $k * n$ channels, which are numbered by $0, 1, \ldots, k * n - 1$. The $k * n$ channels are divided evenly into k subsets: $PC_0, PC_1, \ldots, PC_{k-1}$. The channels belonging to subset PC_x ($x \in \{0, 1, \ldots, k-1\}$) are: $x*n, x*n+1, \ldots, x*n+(n-1)$. Let us assume that channel r is numbered with $x * n + y$ ($y \in \{0, 1, \ldots, n-1\}$). We have $H \gg n$.

C_i assigns priority to each borrowable secondary channel r according to the following rules:

- (1): $priority_r = L$, if $\exists j : j \in IPC_i(r) \wedge p_r(j) = L$; otherwise,
- (2): $priority_r = y + \sum_{j \in IPC_i(r)} p_r(j)$.

C_i selects a secondary channel r with the highest priority from the set of channels which it can borrow, and sends messages to cells C_j ($C_j \in IPC_i(r)$) to borrow channel r. If C_i borrows a channel r successfully, then the channel r can not be used in its primary cells $C_j \in IPC_i(r)$.

Next, we explain the intuition behind assigning priority to available channels. When C_j assigns priority to primary channels included in its reply to C_i, the priorities are assigned in such a way that it encourages C_i to borrow a channel r which has been lent to cells which are not neighbors of C_i, because C_j can not use r anyway. This helps in increasing the reuse of a channel efficiently. At the same time, C_j discourages C_i to attempt to borrow channel r corresponding to which it sent a *grant(r)* message to some C_k where C_k and C_i are neighbors. The goal is to minimize the degree of contention. When multiple cells try to borrow the same channel r, at most one of them will succeed, while others will fail in this try. Thus such a channel r should be given a low priority to be borrowed, decreasing the degree of contention. But in our algorithm a cell C_i still has the opportunity to choose such a channel r to borrow. If C_i is not given a chance to choose r to borrow, then channel utilization may decrease. For example, suppose cell C_j has granted a neighbor C_k to borrow a primary channel r, and C_j receives another neighbor C_i's request for its channel usage information. If C_j does not include r in its reply to C_i, then there is no way that C_i will choose r to borrow. However, the fact that C_j has granted C_k's request to borrow r does not mean that C_k will eventually borrow r successfully. Whether C_k can borrow r successfully depends on response from its other neighbors with r as a primary channel. If C_k fails to borrow r, then excluding r from C_j's reply to C_i denies C_i's opportunity to borrow r, and hence decreasing channel utilization. Note that in such cases, r could have been included in C_j's reply to C_i, i.e., giving C_i an opportunity to choose r to borrow. Under our algorithm, C_j will include r to its reply to C_i in such cases, but C_j informs C_i that r has a low priority to borrow. If C_i chooses r to borrow eventually, it may succeed in borrowing r. Compared with the case that C_k fails to borrow r and all C_k's neighbors are denied the opportunity to choose r to borrow from C_j, our algorithm has better channel utilization.

Next, we explain the idea underlying the rules by which a cell C_i assigns priority to each borrowable secondary channel r. C_i always selects a secondary channel r with the highest priority to borrow. If C_i borrows r successfully, then r can not be used by its primary cells $C_j \in IPC_i(r)$. Let $IPC_i(r)\_BEFORE$ denote the set of cells in $IPC_i(r)$ which can not use channel r before C_i borrows r, and $IPC_i(r)\_AFTER$ denote the set of cells in $IPC_i(r)$ which can not use channel r after C_i borrows r (note that $IPC_i(r)\_AFTER$ is equal to $IPC_i(r)$). The goal of the channel

selection algorithm is to select a channel r in such a way that borrowing r causes less number of cells in $IPC_i(r)$ to become newly unable to use r due to the fact that C_i borrows r. That is, we want to reduce the size of the set $(IPC_i(r)\_AFTER - IPC_i(r)\_BEFORE)$. By doing this, the selection algorithm always chooses a channel to borrow in such a way that it increases the chance of reusing the same channel. Let us illustrate the channel selection algorithm with an example. In Figure 5.1, suppose cell A_4 needs to borrow a channel. After computing the set of channels which it can borrow, suppose it finds out that it can borrow channels $r1$ and $r2$. $r1$ is a primary channel for cells B_1, B_3, B_4, and B_6. $r2$ is a primary channel for cells C_1, C_3, C_4, and C_6. Suppose that $r1$ is an available channel in cells B_1, B_3, B_4, and B_6, while C_1, C_3, and C_4 have either lent $r2$ or granted $r2$ to some other cells which are not interference neighbors of A_4, and C_6 has $r2$ available. According to the channel selection algorithm, cells B_1, B_3, B_4, and B_6 will assign M to $p_{r1}(B_1)$, $p_{r1}(B_3)$, $p_{r1}(B_4)$, and $p_{r1}(B_6)$ respectively. And C_1, C_3, and C_4 will assign H to $p_{r2}(C_1)$, $p_{r2}(C_3)$, and $p_{r2}(C_4)$ respectively, and C_6 will assign M to $p_{r2}(C_6)$. Since $H \gg n$ and $H \gg M$, the value of $priority_r$ is dominated by the value of H. Then the calculated priority of $r2$ has a higher priority than $r1$. Thus, A_4 will select $r2$ to borrow. Suppose A_4 borrows $r2$ successfully. $IPC_i(r2)\_BEFORE := \{C_1, C_3, C_4\}$ and $IPC_i(r2)\_AFTER := \{C_1, C_3, C_4, C_6\}$. Compared with $IPC_i(r2)\_BEFORE$, the number of cells newly added to $IPC_i(r2)\_AFTER$ is 1. Thus only one cell, namely C_6, becomes newly unable to use $r2$, the other cells, C_1, C_3, and C_4, can not use $r2$ even before A_4 borrows $r2$. If A_4 selected $r1$ to borrow, instead of $r2$, and borrowed $r1$ successfully, then $IPC_i(r1)\_BEFORE := \emptyset$ and $IPC_i(r1)\_AFTER := \{B_1, B_3, B_4, B_6\}$. The number of cells newly added to $IPC_i(r1)\_AFTER$ is 4, compared with $IPC_i(r1)\_BEFORE$. Thus 4 cells become unable to use $r1$. From this example, we can see that the channel selection algorithm always selects a channel r in such a way that borrowing r helps in increasing the chance of reusing of r, and therefore improving channel utilization.

5.4.3 Data Structures

The data structures used in the algorithm are given in Table 5.1.

5.4.4 The Algorithm

(A) When C_i needs a channel to support a call, it computes the set of channels that are free,

namely, $Free_i$. If $Free_i = \emptyset$, then C_i sets a timer and sends a *request* message to each cell $C_j \in INb_i$. Else, a channel $r \in Free_i$ is picked to support the call and added to U_i. When the call terminates, r is deleted from U_i.

(B) When C_i receives a *request* message from C_j, it computes R_i. If $R_i \neq \emptyset$, then sends $reply(R_i)$ to C_j; else discards the *request* message.

(C) After C_i gets *reply* message from all its interference neighbors or the timer expires, it sets a new timer, sets $Avail_i := \emptyset$, and does the following.

(C.1) $\forall\ r \in Spectrum$, $Avail_i := Avail_i\ \cup\ \{r\}$ if the following two conditions are satisfied:

1: $r \notin U_i$ (i.e., r is not being used by C_i);

2: $\forall C_j \in IPC_i(r)$, C_i got $reply(R_j)$ *and* $r \in R_j$.

(C.2) If $Avail_i \neq \emptyset$, then C_i chooses a channel $r \in Avail_i$ as per the channel selection algorithm and sends a *transfer(r)* message to all cells in $IPC_i(r)$. Otherwise, the call is dropped.

(D) When C_i receives a *transfer(r)* message from cell C_j,

(D.1) It computes $Free_i$. If $r \in Free_i$, then C_i sends a *grant(r)* message to C_j and adds C_j to $Grant_i(r)$.

(D.2) Else if $r \in U_i$ or $Lent_i(r)\ \cap\ INb_j \neq \emptyset$, then C_i sends a *refuse(r)* message to C_j.

(D.3) Else let $S := Grant_i(r)\ \cap\ INb_j$. If $S = \emptyset$, then C_i sends a *grant(r)* message to C_j and adds C_j to $Grant_i(r)$.

(D.4) Else if $\forall C_k \in S$, the timestamp of C_j's request is less than that of C_k's request, then C_i sends a *conditional_grant(S,r)* message to C_j and adds C_j to the set $Grant_i(r)$. Otherwise, C_i sends a *refuse(r)* message to C_j.

(E) If C_i receives responses to its *transfer(r)* message from each cell in $IPC_i(r)$ before the timer set in step (C) expires, it checks for the following three conditions:

(E.1) each response is either a *grant(r)* message or a *conditional_grant(S,r)* message;

(E.2) there is at least one *grant(r)* message;

(E.3) $\forall$ *conditional_grant*(S, r) and $\forall\ C_j \in S$, a *grant(r)* message from some cell C_k has been received by C_i where $C_k \in (IPC_i(r)\ \cap\ IPC_j(r))$.

If all the conditions E.1, E.2 and E.3 are met, then C_i sends a *use(r)* message to each $C_j \in IPC_i(r)$ and uses channel r to support the call. r is added to U_i. When the call finishes, C_i removes r from U_i and sends a *release(r)* message to each $C_j \in IPC_i(r)$. Otherwise(not all conditions E.1, E.2 and E.3 are met or the timer set in step (C) expires), it sends an *abort(r)* message to each cell in $IPC_i(r)$ from which a *grant(r)* message or a *conditional_grant(S,r)* message is received. r is deleted from $Avail_i$. If the timer set in step (C) does not expire, then executes step(C.2); else drops the call.

(F) When a cell C_i receives an *abort(r)* message from C_j, it deletes C_j from $Grant_i(r)$. When a cell C_i receives a *use(r)* message from C_j, it deletes C_j from $Grant_i(r)$ and adds C_j to $Lent_i(r)$. When C_i receives a *release(r)* message from C_j, it deletes C_j from $Lent_i(r)$.

5.4.5 Proof of Correctness of the Algorithm

Theorem 5.4.1 *Under the proposed algorithm, no two interference neighbors are allowed to use the same channel concurrently.*

Proof: Suppose that two cells C_i and C_j are in each other's interference neighborhood, and they are using the same channel r concurrently. They cannot both be the primary cells of channel r since they are in each other's interference neighborhood. So at most one of them is a primary cell of r. There are three possibilities.

1: **Both C_i and C_j are not primary cells of channel r.**

This means both C_i and C_j have borrowed r. Without loss of generality, assume that C_i's *request* has a smaller timestamp than that of C_j's *request*. Then, $IPC_i(r) \cap IPC_j(r) \neq \emptyset$ (Property 5.2.2). Let $C_k \in IPC_i(r) \cap IPC_j(r)$. There are two cases.

- C_k **sends a *grant(r)* message to C_i first.** In this case, when C_k receives C_j's *transfer(r)* message, a *refuse(r)* message is sent to C_j, since C_j's request has a higher timestamp than C_i's. So C_j would not have been able to borrow channel r, which is a contradiction to our assumption that C_i and C_j use r concurrently.

- C_k **sends a *grant(r)* message to C_j first.** In this case, when C_k receives C_i's request, it will send a *conditional_grant(S,r)* message to C_i, according to step (D.4) in the algorithm. If C_j got *grant(r)* messages from each cell in $IPC_j(r)$, then C_i would not have been able to acquire r because it could not have got a *grant(r)* message from any cell in $IPC_i(r) \cap IPC_j(r)$, in which case it would have failed to meet condition (E.3). If C_i received a *grant(r)* message from any cell $C_m \in IPC_i(r) \cap IPC_j(r)$, then C_m would have sent a *refuse(r)* message to C_j, so C_j could not have acquired r, which is a contradiction to our assumption.

So C_i and C_j will not use the same channel r concurrently in this case.

2: C_i **is a primary cell of** r**,** C_j **is a secondary cell of** r**.**

In this case, C_i must have acquired r to support a call in its own cell and also must have lent it to C_j. This is not possible because a cell lends a primary channel to a neighbor only if it is not using this channel (see step (D.2) of the Algorithm). Moreover a cell does not use a primary channel to support a call if this channel has been lent to an interference neighbor (see step (A) of the Algorithm).

3: C_j **is a primary cell of** r**,** C_i **is a secondary cell of** r**.**

This case is similar to (2). □

Theorem 5.4.2 *The algorithm is deadlock-free.*

Proof: In the channel allocation algorithm, a timeout strategy is used. A cell sets a timer when it sends a *request* message or a *transfer(r)* (where r is the selected channel to borrow) message. It starts to proceed either after it receives responses corresponding to each of its message (a *request* or *transfer(r)*) or after the timer expires. So hold and wait situation does not occur and hence, the algorithm is deadlock-free. □

5.5 Performance of the Algorithm

In this section, we compare the performance of our algorithm with that of the algorithm proposed in [6]. We choose the algorithm proposed in [6] for comparison because both our algorithm and the algorithm in [6] are fault-tolerant. Moreover, both of them adopt the Resource Planning Model and a Search approach. Some of the data structures used in [6] are shown in Table 5.2. The *agree*

and *conditional_agree* messages in [6] are equivalent to the *grant* and *conditional_grant* messages in our algorithm respectively.

There are six main differences between our algorithm and the algorithm proposed in [6]. Let r be a primary channel of cell C_i.

- 1: In [6], in cell C_i, no effort is made to distinguish the set of cells which have borrowed channel r successfully from the set of cells which are attempting to borrow r. All potential borrowers are maintained in one set, namely $I_i(r)$, including both the cells which have already acquired channel r successfully and the cells which are trying to borrowing channel r. In our algorithm, we maintain two sets $Grant_i(r)$ and $Lent_i(r)$. $Grant_i(r)$ refers to the set of borrowers which try to borrow channel r, but have not succeeded yet, while $Lent_i(r)$ is the set of borrowers which have already acquired channel r successfully.

- 2: In [6], after a neighbor C_j borrows channel r from C_i successfully, C_j does not notify C_i about this, while C_j explicitly notifies C_i about this in our algorithm.

- 3: Different way of computing set R_i which is attached to C_i's *reply* message to C_j. In [6], a primary channel r is not included into R_i if $I_i(r) \cap IN_j \neq \emptyset$. This happens even when no cell in $I_i(r)$ has really acquired r at the time when R_i is computed. Doing so makes the size of R_i smaller than it should be. The cells in $I_i(r)$ may not acquire r successfully eventually, in which case C_j should be given a chance to borrow r. In our algorithm, channel r can be included into R_i as long as C_i is not using r and r has not been lent to any interference neighbors of cell C_j.

- 4: Different way of handling a *request* message from cell C_j. In [6], C_i always sends *reply* message to C_j upon receiving C_j's *request*, while no *reply* message is sent if $R_i = \emptyset$ in our algorithm.

- 5: Different way of handling a *transfer(r)* message from cell C_j. In [6], C_i sends *conditional_agree(S,r)* to C_j if $(I_i(r) - CI_i(r,j)) \cap IN_j \neq \emptyset$ and C_j's *request* has the smallest timestamp. In our algorithm, a *refuse* message will be sent to C_j if $Lent_i(r) \cap INb_j \neq \emptyset$.

- 6: Different channel selection algorithm. In [6], a cell C_i always borrows a channel from its

Table 5.1: Data structures used in our algorithm at each cell C_i

$Spectrum$:	all the channels in the system.
INb_i:	defined before.
$IPC_i(r)$:	defined before.
$Avail_i$:	the set of channels C_i may borrow.
PC_i:	the set of channels pre-allocated to cell C_i.
U_i:	the set of channels being used at cell C_i. Initially $U_i := \emptyset$.
R_i:	the set of channels C_i attaches to its *reply* message in response to a *request* message from a neighbor C_j. $R_i := PC_i - U_i - \{r \mid r \in PC_i \wedge \ (Lent_i(r) \cap (INb_j \cup C_j) \neq \emptyset)\}$ where $INb_j := \{C_k \mid dist(C_j, C_k) < D_{min}\}$.
$Grant_i(r)$:	the set of cells to which C_i has sent a *grant(r)* or *conditional_grant(S,r)* where S is a set of cells computed dynamically. Initially, $Grant_i(r) := \emptyset$.
$Lent_i(r)$:	the set of cells from which cell C_i has received a *use(r)* message, i.e., the set of cells to which r has been lent. Initially $Lent_i(r) := \emptyset$.
$Free_i$:	the set of primary channels r of cell C_i such that C_i is not using r and $Grant_i(r)$ and $Lent_i(r)$ are empty. That is: $Free_i := PC_i - U_i - \{r \mid r \in PC_i \wedge ((Grant_i(r) \cup Lent_i(r)) \neq \emptyset)\}$.

Table 5.2: Data structures used in the algorithm proposed in [6]

P_i:	the set of primary channels assigned to C_i.
U_i:	the set of channels being used in C_i. Initially U_i is an empty set.
$I_i(r)$:	the set of cells to which C_i has sent an *agree(r)* message. Initially $I_i(r)$ is an empty set. If $I_i(r) \neq \emptyset$, then r is an interference channel of cell C_i. C_i can not use r, but it can lend r to other cells.
$CI_i(r, j)$:	a set of cells, which saves the state of $I_i(r)$ when C_i receives C_j's *request* message.
IN_i:	the set of interference neighbors of cell C_i.

richest neighbors. The richness of a neighbor C_j ($C_j \in S_x \cap INb_i, C_i \notin S_x$) is defined as the minimum number of primary channels which are available in the interference primary cells of PC_x. The goal of borrowing a channel from richest neighbors is to reduce the the probability of the lender running out of channels. The interference caused by borrowing a channel to the cells which have the channel allocated to them is not taken into account in this algorithm. In our algorithm, when a cell C_i tries to select a channel to borrow, for each secondary channel r that it may borrow, it computes the set of cells in $IPC_i(r)$ which can not use channel r before it borrows r, this set is denoted by $IPC_i(r)\_BEFORE$. It also computes the cells in the set $(IPC_i(r) - IPC_i(r)\_BEFORE)$ which become unable to use r due to the fact that C_i borrows r. The goal is to reduce the size of this set, that is, to reduce $|IPC_i(r) - IPC_i(r)\_BEFORE|$. By reducing the size of this set, channel utilization is improved.

The first three differences make the set of channels which are included in the *reply* message larger in our algorithm than in [6] because $|Lent_i(r)| \leq |I_i(r)|$. This makes our algorithm make better use of the set of available channels. The fourth difference makes our algorithm to reduce message complexity and save some bandwidth. It is of no use for C_i to send a reply message with an empty set of available channels to a neighbor C_j. Doing this increases the message complexity and wastes bandwidth. The fifth difference helps in acquiring a channel faster in our algorithm. The last difference enables our algorithm to reduce the interference caused by borrowing a channel to the cells which have the channel allocated to them, so improving channel utilization.

To help understand the differences between our algorithm and the algorithm proposed in [6], suppose r is a primary channel of A_4 in Figure 5.1. $I_{A_4}(r) := \emptyset$ in [6] and $Grant_{A_4}(r) := \emptyset \wedge Lent_{A_4}(r) := \emptyset$ in our algorithm.

- Suppose A_4 first sends a *grant(r)* (*agree(r)* in [6]) message to E_3 and adds E_3 to $Grant_{A_4}(r)$ ($I_{A_4}(r)$ in [6]). Then it receives a *transfer(r)* message from I_3. In [6], r will not be included into R_i to cell I_3 since E_3 is an interference neighbor of I_3 and $E_3 \in I_{A_4}(r)$. In our algorithm, r will be included into R_i since r is not being used by either A_4 or any cell which is an interference neighbor of cell I_3. Thus under our algorithm, the set of channels included in R_i is larger than that in [6] which results in better channel usage.

- Suppose A_4 receives *request* messages from both E_3 and I_3, r is included in the *reply* message to both of them. E_3 first acquires r and A_4 adds E_3 to $Lent_{A_4}(r)$ ($I_{A_4}(r)$ in [6]). Then A_4 receives I_3's *transfer(r)* message. If I_3's *request* has a smaller timestamp than that of E_3's, then a *conditional_agree* message will be sent to I_3 in [6]. I_3 can never acquire channel r because r is being used by E_3 which is an interference neighbor of I_3[3]. But I_3 will still try to check (E.3) if it meets (E.1) and (E.2). This makes both A_4 and I_3 to waste time to compute. In our algorithm, A_4 will send a *refuse* message to I_3 because A_4 knows that E_3 is now using r and E_3 is an interference neighbor of I_3. I_3 will not try to check (E.3) because it receives a *refuse* message. Thus both A_4 and I_3 make decision faster in our algorithm than in [6] in this case.

- Now, suppose A_4 wants to borrow a channel from neighbors, and cells B_1, B_3, B_4, and B_6 are its richest neighbors. Suppose r is a common available primary channel in B_1, B_3, B_4, and B_6, and r is borrowed by A_4 according to the channel selection algorithm in [6]. Then all these lenders of channel r can not use r. The number of cells in $\{B_1, B_3, B_4, B_6\}$ that become unable to use r due to the fact that A_4 borrows r is 4. In our algorithm, suppose r is a primary channel of cells I_0, I_1, and I_3. Suppose I_0 has lent r to C_0, I_1 has lent r to B_2, and I_3 has lent r to E_6. Suppose according to our channel selection algorithm, r has the highest priority to borrow, then A_4 borrows r. The number of cells in $\{I_0, I_1, I_3\}$ that become unable to use r due to the fact that A_4 borrows r is 0. Thus, the (newly added) interference caused by borrowing channel r is reduced. Therefore, our channel selection algorithm helps improve channel utilization.

5.6 Simulation Results

In this section, we compare the performance of our algorithm with that of the algorithm proposed in [6] in terms of call failure rate under non-uniform traffic pattern. Call failure rate is defined as the ratio of total number of calls dropped (including dropped new calls originating in the cell and dropped inter-handoff calls) to the total number of calls processed. Call arrival rate is defined as the number of call arrivals per hour per cell.

[3] I_3 can not meet condition (E.3).

5.6.1 Simulation Parameters

The simulated cellular network consists of $9 * 9$ cells (see Figure 5.1). Each cell has 30 neighbors (by wrapping around the cells). Totally the number of channels in the system is $44 * 9$, each cell being allocated 44 primary channels. We assume that the average one-way communication delay between two cells is 2 milliseconds when network congestion is not present. This average delay includes transmission delay, propagation delay and the message processing time, which is the same as that in [6]. We assume that the maximum delay an inter-handoff call can tolerate is 10.0 milliseconds which is the same as in [6]. In the simulation, a large number of mobile hosts are generated. After generation, a mobile host sends call requests to the base station in its cell. It may move from one cell to an adjacent cell while involved in a communication. We assume that the average service time per call is 3 minutes, which is the same as used in [6].

Under non-uniform traffic pattern, a cell can be in one of the two states, namely, normal or hot. When a cell is in normal state, call arrival rate is low, and handoff rate is high; when a cell is in hot state it has high call arrival rate and low handoff rate. The parameters used for non-uniform traffic pattern are given in Table 5.3.

Table 5.3: Parameters for non-uniform traffic pattern

Mean call arrival rate in a normal cell	λ
Mean call 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 changing from normal to hot state	$1/1800s$
Mean rate changing from hot to normal state	$1/180s$
Mean service time per call	$180s$

5.6.2 Simulation Results

All the comparisons between our algorithm and the algorithm proposed in [6] are made under non-uniform traffic pattern. First we compare the performance of the two algorithms in the absence of cell failures. Figure 5.3 shows that under non-uniform traffic pattern, the failure rate of our algorithm is smaller than that of the algorithm proposed in [6]. The reason is that in our algorithm, the size of the set of channels in the reply message is larger than that in [6], and

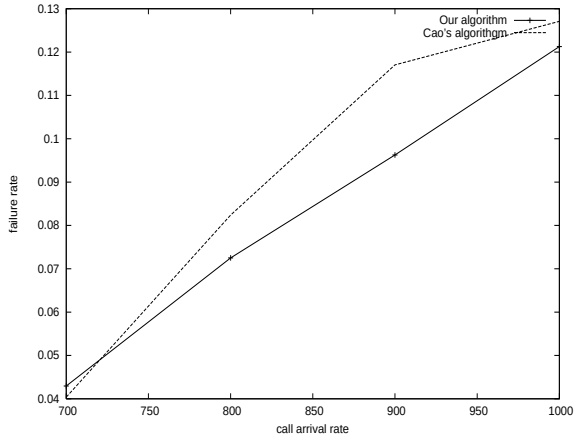


Figure 5.3: Performance without cell failure

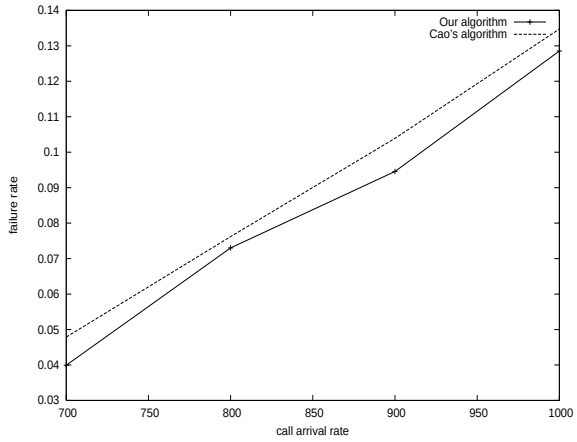


Figure 5.4: Performance with one cell failure

the channel selection algorithm used in our channel allocation algorithm reduces the interference caused by borrowing a channel, thus channels are reused in a more efficient way, making less calls to be dropped.

Figure 5.4 and Figure 5.5 show performance comparison of our algorithm with that proposed in [6] under non-uniform traffic pattern in the presence of one and two cell failures. The simulation results show that call failure rate under our algorithm is lower than that of [6] under non-uniform traffic pattern.

5.7 Conclusion

Existing distributed fault-tolerant channel allocation algorithms do not make full use of the available channels. In this chapter, we proposed a fault-tolerant algorithm for channel allocation which makes efficient reuse of channels. Under our algorithm, a cell that tries to borrow a channel does

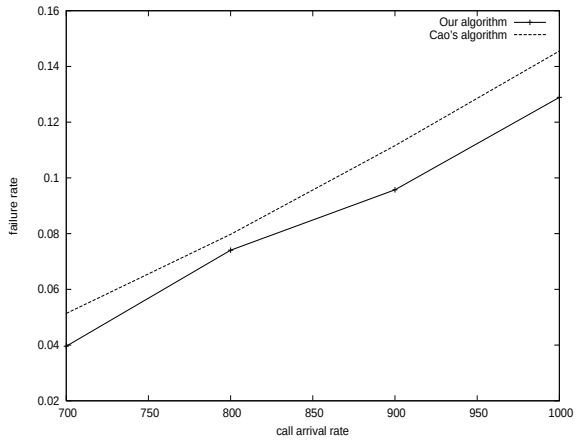


Figure 5.5: Performance with two cell failures

not have to wait until it receives channel usage information from each of its interference neighbors. A cell can borrow a channel as long as it receives channel usage information from each cell in a subgroup in its interference neighborhood and there is at least one common primary channel which is not being used by any cell in this subgroup, which makes the algorithm fault-tolerant. Moreover, the channel selection algorithm used in the channel allocation algorithm takes into account the interference caused by borrowing a channel. A cell C_i chooses a channel r to borrow such that the number of neighbors that become newly unable to use r due to the fact that C_i borrows r is reduced. By doing this, channel utilization is improved.

Chapter 6

Future Work

The main service supported by the current cellular networks is voice service, as well as low-bit-rate data services. With the advance in cellular telecommunication technology, the services that people want cellular networks to support also are on the increase. The future cellular networks are expected to offer more and better services: high speed wireless Internet access and wireless multimedia services, including audio, video, images, and data [25]. These new services have new requirements on channel allocation. For example, wireless multimedia services ask for varied length channel spectrum and they are time constrained. Quality of service (QoS) should be guaranteed for these services. Thus new channel allocation algorithms need to be proposed to meet these new demands.

Some research has been done to support QoS in multimedia wireless networks [31, 26, 14]. In [31], the authors proposed an admission control scheme to provide QoS guarantees for multimedia traffic carried in high-speed wireless cellular networks. The scheme is based on adaptive bandwidth reservation. A connection is allowed and supported in a cell only if the cell can allocate enough bandwidth for this connection and enough bandwidth can be reserved in all the neighboring cells. When a mobile node moves from one cell to another during a connection, bandwidth is allocated in the new cell and is reserved in all the neighboring cells of the new cell, and the reserved bandwidth in more distant cells is released. Although this scheme may guarantee QoS to the existing connections, it is, in many cases, too conservative and pessimistic. In [26, 14], schemes that support QoS in multimedia application in high-speed wireless network were proposed. It is indicated in these papers that multimedia applications could tolerate and adapt gracefully to transient fluctuations in the QoS that they receive from the network. By exploiting this feature of

multimedia applications, algorithms were proposed to allow bandwidth to be borrowed temporarily from the existing connections to support a newly admitted connection. The algorithm guarantees that no connection gives up more than its fair share of bandwidth and the borrowed bandwidth is returned to the degraded connection as soon as possible. Thus the bandwidth is used efficiently and managed in a fair manner.

It is very important to design algorithms to allocate bandwidth efficiently and to guarantee QoS for multimedia applications and other applications in high-speed wireless networks. It is better for the algorithm to adapt to the current traffic load and bandwidth are reserved and allocated in an adaptive way, instead of in a fixed manner. It is also desirable if bandwidth can flow from lightly loaded cells to heavily loaded ones. Moreover, handoff is an even challenging issue in multimedia applications in wireless networks. From the point view of user, it is less desirable to drop a handoff connection than to drop a new connection.

In the future, we want to put effort in designing algorithms which can allocate bandwidth and guarantee QoS to support multimedia application as well as non-multimedia applications in high-speed wireless network.

Bibliography

[1] Dharma Prakash Agrawal and Qing-An Zeng. *Introduction to Wireless and Mobile Systems*. Brooks/Cole Thomsom, 2003.

[2] 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.*, 44:14–23, Feb 1995.

[3] Azzedine Boukerche, Sungbum Hong, and Tom Jacob. A Performance Study of a Distributed Algorithm for Dynamic Channel Allocation. *ACM MSWiM 2000 (Modeling, Analysis and Simulation of Wireless and Mobile Systems)*, pages 36–43, August 2000.

[4] Azzedine Boukerche, Sungbum Hong, and Tom Jacob. A Distributed Algorithm for Dynamic Channel Allocation. *Mobile Networks and Applications*, 7:115–126, 2002.

[5] Guohong Cao and Mukesh Singhal. An Adaptive Distributed Channel Allocation Strategy for Mobile Cellular Networks. *Journal of Parallel and Distributed Computing, special issue on Mobile Computing*, 60(4):451–473, April 2000.

[6] Guohong Cao and Mukesh Singhal. Distributed Fault-Tolerant Channel Allocation for Cellular Networks. *IEEE journal on selected areas in communications*, 18(7):1326–1337, JULY 2000.

[7] Guohong Cao and Mukesh Singhal. Efficient Distributed Channel Allocation for Mobile Cellular Networks. *Computer Communications*, 23(10):950–961, May 2000.

[8] Manhoi Choy and Ambuj K. Singh. Efficient Distributed Algorithms for Dynamic Channel Assignment. In *Proceedings of 7th IEEE International Symp. on Personal, Indoor and Mobile Radio Communication*, 1996.

[9] J. C.-I. Chuang. Performance Issues and Algorithms for Dynamic Channel Assignment. *IEEE Journal on Selected Areas in Communications*, 11(6):955–963, August 1993.

[10] Sajal K. Das, Sanjoy K. Sen, and Rajeev Jayaram. A Dynamic Load Balancing Strategy for Channel Assignment Using Selective Borrowing in Cellular Mobile Environment. *Wireless Networks*, 3:333–347, 1997.

[11] Sajal K. Das, Sanjoy K. Sen, and Rajeev Jayaram. A Structured Channel Borrowing Scheme for Dynamic Load Balancing in Cellular Networks. In *Proceedings of the 17th International Conference on Distributed Computing Systems (ICDCS'97)*, pages 116–123, 1997.

[12] X. Dong and T. H. Lai. An Efficient Priority-Based Dynamic Channel Allocation for Mobile Cellular Networks. In *IEEE INFOCOM*, pages 892–899, 1997.

[13] X. Dong and T. H. Lai. Distributed Dynamic Carrier Allocation in Mobile Cellular Networks: Search vs. Update. In *Proceedings of the 17th International Conference on Distributed Computing Systems(ICDCS'97)*, pages 108–115, May 1997.

[14] Mona EI-Kadi, Stephan Olariu, and Hussein Abdel-Wahab. A Rate-Based Borrowing Scheme for QoS Provisioning in Multimedia Wireless Networks. *IEEE Transactions on Parallel and Distributed Systems*, 13(2):156–166, February 2002.

[15] S. A. El-Dolil, W. C. Wong, and R. Steele. Teletraffic Performance of Highway Microcells with Overlay Macrocell. *IEEE Journal on Selected Areas in Communications*, 7:71–78, January 1989.

[16] P.-O. Gaasvik, M. Cornefjord, and V. Svensson. Different Methods of Giving Priority to Handoff Traffic in a Mobile Telephone System with Directed Retry. In *41st IEEE Vehicular Technology Conference, Gateway to the Future Technology in Motion*, pages 549–553, 1991.

[17] Samrat Ganguly, Badri Nath, and Navin Goya. Optimal Bandwidth Reservation Schedule in Cellular Network. In *Proceedings of the IEEE/INFOCOM 2003, San Francisco, CA*, April 2003.

[18] D.J. Goodman. Cellular Packet Communication. *IEEE Trans. Commun.*, 38:1272–1280, Aug 1990.

[19] J. Ioannidis, D. Duchamp, and G. Q. Maguire. IP-Based Protocols for Mobile Internetworking. In *Proceedings of the ACM SIGCOMM Symposium on Communication, Architectures and Protocols*, pages 235–245, September 1991.

[20] Jianping Jiang, Ten-Hwang Lai, and Neelam Soundarajan. On Distributed Dynamic Channel Allocation in Mobile Cellular Networks. *IEEE Transactions on Parallel and Distributed Systems*, 13(10):1024–1037, October 2002.

[21] K. Kyamakya and K. Jobmann. Location management in cellular networks: classification of the most important paradigms, realistic Simulation framework, and relative performance analysis. *IEEE Transactions on Vehicular Technology*, 54(2):687–708, March 2005.

[22] Ten-Hwang. Lai, Jianping Jiang, and Tao Ma. A Relaxed Mutual Exclusion Problem with Application to Channel Allocation in Mobile Cellular Networks. In *Proceedings of the 20th International Conference on Distributed Computing Systems(ICDCS)*, pages 592–599, 2000.

[23] L. Lamport. Time, Clocks and The Ordering of Events in a Distributed System. *Commun. ACM*, 21(7):558–565, July 1978.

[24] W. C. Y. Lee. New Cellular Schemes for Spectral Efficiency. *IEEE Transactions on Vehicular Technology*, VT-36, November 1987.

[25] Yi-Bing Lin and Imrich Chlamtac. *Wireless and Mobile Network Architectures*. John Wiley and Sons, Inc., 2001.

[26] Anjlica Malla, Mona EI-Kadi, Stephan Olariu, and Petia Todorova. A Fair Resource Allocation Protocol for Multimedia Wireless Networks. *IEEE Transactions on Parallel and Distributed Systems*, 14(1):63–71, January 2003.

[27] R. Mathar and J. Mattfeldt. Channel Assignment in Cellular Radio Networks. *IEEE Transactions on Vehicular Technology*, 42(4):647–656, November 1993.

[28] S. Nanda and D. J. Goodman. Dynamic Resource Acquisition: Distributed Carrier Allocation for TDMA Cellular Systems. In *Proc. GOLBECOM'91*, pages 883–889, Dec 1991.

[29] Sanket Nesargi and Ravi Prakash. Distributed Wireless Channel Allocation in Networks with Mobile Base Stations. In *INFOCOM (2)*, pages 592–600, 1999.

[30] Sanket Nesargi and Ravi Prakash. Distributed Wireless Channel Allocation in Networks with Mobile Base Stations. *IEEE Transactions on Vehicular Technology*, 51(6):1407–1421, November 2002.

[31] Carlos Olieira, Jaime Bae Kim, and Tatsuya Suda. An Adaptive Bandwidth Reservation Scheme for High-Speed Multimedia Wireless Networks. *IEEE Journal on Selected Areas in Communications*, 16(6):858–874, August 1998.

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

[33] Ravi Prakash, Niranjan G. Shivaratri, and Mukesh Singhal. Distributed Dynamic Fault-Tolerant Channel Allocation for Cellular Networks. *IEEE Transactions on Vehicular Technology*, 48(6):1874–1888, November 1999.

[34] Ravi Prakash and Mukesh Singhal. Distributed Wireless Channel Allocation in Cellular Systems with Mobile Base Stations. In *Workshop on Nomadic Computing(satellite workshop of IPPS), Geneva*, April, 1997.

[35] Mischa Schwartz. *Mobile Wireless Communications*. Cambridge University Press, 2005.

[36] Mukesh Singhal and Niranjan G. Shivaratri. *Advanced Concepts in Operating Systems: Distributed, Database, and Multiprocessor Operating Systems*. McGraw-Hill, Inc, 1994.

[37] T. Tugcu, I.F. Akyildiz, and E. Ekici. Location Management Framework for Next Generation Wireless Systems. In *Proceedings of the IEEE ICC'2004 Conference, Paris, France*, June 2004.

[38] Jianchang Yang, Q. Jiang, D. Manivannan, and M. Singhal. A Fault-Tolerant Distributed Channel Allocation Scheme for Cellular Networks. *IEEE Transactions on Computers*, 54(5):616–629, 2005.

[39] Jianchang Yang, Qiangfeng Jiang, and D. Manivannan. A Fault-Tolerant Channel Allocation Algorithm for Cellular Networks with Mobile Base Stations. *Accepted for publication in IEEE Transactions on Vehicular Technology*.

[40] Jianchang Yang and D. Manivannan. An Efficient Fault-Tolerant Distributed Channel Allocation Algorithm for Mobile Computing Systems. In *Proceedings of the Third International Conference on Parallel and Distributed Computing, Applications, and Technologies (PDCAT2002), Sept.4-6, 2002, Kanazawa, Japan*, pages 372–377.

[41] Jianchang Yang and D. Manivannan. A Fault-Tolerant Channel Allocation Algorithm for Cellular Networks with Mobile Base Stations. In *Proceedings of the 2003 International Conference on Wireless Networks(ICWN'03), Las Vegas, Nevada, USA, (CSREA Press) June 23-26*, pages 146–152, 2003.

[42] Jianchang Yang and D. Manivannan. An Efficient Fault-Tolerant Distributed Channel Allocation Algorithm for Cellular Networks. *IEEE Transactions on Mobile Computing*, 4(6):578–587, 2005.

[43] Jianchang Yang, D. Manivannan, and Mukesh Singhal. A Fault-Tolerant Dynamic Channel Allocation Scheme for Enhancing QoS in Cellular Networks. In *IEEE Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS-36), Big Island of Hawaii, January 6-9, 2003*, pages 306–315.

[44] M. Zhang and T.-S. P. Yum. Comparisons of Channel-Assignment Strategies in Cellular Mobile Telephones Systems. *IEEE Transactions on Vehicular Technology*, 38(4):211–215, November 1989.

[45] M. Zhang and T.-S. P. Yum. The Nonuniform Compact Pattern Allocation Algorithm for Cellular Mobile Systems. *IEEE Transaction on Vehicular Technology*, 40(2):387–391, May 1991.

[46] Y. Zhang, S. K. Das, and X. Jia. D-CAT : An Efficient Approach for Distributed Channel Allocation in Cellular Mobile Networks . *ACM Mobile Networks and Applications (Special Issue on Parallel Processing Applications in Mobile Computing, Guest Eds: M. Kumar and A. Zomaya)*, 9(4):279–288, July, 2004.

[47] Y. Zhang, X. Jia, , and S. K. Das. An Efficient Approach for Distributed Channel Allocation for Cellular Mobile Networks. In *Proceedings of the Fifth International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M), Rome, Italy*, pages 87–94, July, 2001.

Vita

Jianchang Yang
Date of Birth: 12/30/1973
Place of Birth: Hebei Province, China

- **EDUCATION**
  - B.S. in Medicine, May 1996. Capital University of Medical Science, Beijing, P.R. China.
- **TEACHING INTERESTS**
  - program design and development, microcomputer applications
  - operating systems, distributed operating systems
  - networking core technologies, database development
  - discrete mathematics, data structures, and algorithm
  - Visual Basic, Javascript, C++, Java, C#, perl, php, mysql
  - web page development, data-driven web design, Internet Servers administration
- **PROJECTS**
  - Command Interpreter: This project implements a command interpreter, and it is a simplified version of shell. It gets commands from user and executes them. It supports I/O redirection, execution in background, and pipe.
  - Process Synchronization: This project creates and maintains a Student database. Both students and their advisors can access the database. The shard database is created and loaded into a shared memory. Semaphores are used to synchronize accesses to the database.
  - Database: This project creates a sql database which stores and maintains information for a food distribution center. Multiple tables are created and the relationship between them is set up properly. Some tables are: individual, family, child, infant, adults, agency, request, and allocate.
  - Image Database: This project creates and manages an image database in Object Store. It supports some basic operations, such as insert an image, delete an image, describe an image and search an image. The system provides a GUI interface to the users using java programming language.
  - HTTP Server and Client: This project implements a simplified version of HTTP server and client (yet the core part of the HTTP protocol). The HTTP Client gets a URL from a user and interacts with the HTTP server using the HTTP protocol. The Server receives requests from Client and sends response to Client.
  - SMTP Client: This project implement a simple SMTP client in C, using basic socket programming interfaces. The simple SMTP client includes the core part of the SMTP protocol. The client supports the function of attachment.

- DNS Client: This project implements a simple DNS client in C. The client implements the DNS protocol to interact with DNS server. The client gets an input from the user, makes an query packet, and sends it to the DNS server using UDP. It then waits for a response from the DNS server, interpret the response, and prints out the output.
- Simple Stateless Network File Server: This project implements a simple stateless network file server. It supports remote file service model. It is implemented by using Sun RPC. The server supports the following common operations on files: create, read, write, list, copy, and delete.
- Simple Network Router: This project implements basic functions of a network router. It listens to a particular socket, and waits for network packets to arrive. Once a packet arrives, the simple router will process this packet by looking at its destination address. It looks up the routing table and decides which output interface this packet should be sent to. It also prints this information. This simple router supports packet fragmentation and reassemble.

• PROFESSIONAL SKILLS

- Programming languages: Visual Basic, Javascript, C/C++, perl, Java, php, html, mysql
- database development, web page development, data-driven web design
- Operating Systems: Unix, Solaris, Linux, Windows XP/NT/9x
- Software: Microsoft Office, Visual Studio
- Network Programming: TCP/IP, glomosim
- Computer simulations: csim, glomosim

• RESEARCH INTERESTS

- **Mobile Computing Systems:** Channel allocation and fault tolerance in mobile computing systems.
- **Distributed Systems:** Checkpointing and recovery in distributed computing systems and mobile computing systems.
- **Ad Hoc Networks:** Energy efficient routing and broadcasting in ad hoc networks.
- **Bioinformatics:** Inferring a protein's shape and function from a sequence of amino acids.

- **PUBLICATIONS**
  - **Journal papers:**
    - **J. Yang**, Q. Jiang, D. Manivannan and M. Singhal. "A Fault-Tolerant Distributed Channel Allocation Scheme for Cellular Networks". *IEEE Transaction on Computers* 2005 volume 54, number 5, pages: 616-629.
    - **J. Yang** and D. Manivannan. "An Efficient Fault-Tolerant Distributed Channel Allocation Algorithm for Cellular Networks". *IEEE Transactions on Mobile Computing* 2005 volume 4, number 6, pages: 578-587.
    - **Jianchang Yang**, Qiangfeng Jiang, and D. Manivannan. "A Fault-Tolerant Channel Allocation Algorithm for Cellular Networks with Mobile Base Stations". Accepted by IEEE Transactions on Vehicular Technology.
    - **Jianchang Yang** and D. Manivannan. "Performance Comparison of Two Channel Allocation Approaches: Channel Pre-allocation Vs. Non-Pre-allocation". submitted to IEEE Transactions on Vehicular Technology.
    - D. Manivannan, Q. Jiang, **J. Yang** and M. Singhal. "Asynchronous Recovery based on Staggered Quasi-Synchronous Checkpointing". submitted to International Journal of Information Science and Engineering.
  - **Conference publications:**
    - **Jianchang Yang** and D. Manivannan. "A Fault-Tolerant Channel Allocation Algorithm for Cellular Networks with Mobile Base Stations". In Proceedings of the 2003 International Conference on Wireless Networks (ICWN'03), Las Vegas, Nevada, USA, (CSREA Press) June 23-26, 2003, pages 146-152.
    - **Jianchang Yang**, D. Manivannan and Mukesh Singhal. "A Fault-Tolerant Dynamic Channel Allocation Scheme for Enhancing QoS in Cellular Networks" In IEEE Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS-36), Big Island of Hawaii, January 6-9, 2003.
    - **Jianchang Yang** and D. Manivannan. "An Efficient Fault-Tolerant Distributed Channel Allocation Algorithm for Mobile Computing Systems" In Proceedings of the Third International Conference on Parallel and Distributed Computing, Applications, and Technologies (PDCAT 2002), Sept. 4-6, 2002, Kanazawa, Japan, pages 372-377.
    - D. Manivannan, Q. Jiang, **J. Yang**, and M. Singhal. "Asynchronous Recovery based on Staggered Quasi-Synchronous Checkpointing", International Workshop on Distributed Computing (IWDC) 2005, pages 117-128.

- **PROFESSIONAL ACTIVITIES**
  - served as a reviewer for WILEY'S WIRELESS COMMUNICATIONS AND MOBILE COMPUTING JOURNAL and Journal of Parallel and Distributed Computing.

- **REFERENCES**
  - Dr. D. Manivannan, Associate Professor
    Department of Computer Science, University of Kentucky
    231 James F. Hardymon Building, 301 Rose Street
    Lexington, KY 40506-0495
    Phone: (859) 257-9234, Fax: (859) 323-3740
    E-Mail: mani@cs.uky.edu

– Dr. Mukesh Singhal, Professor and Gartner Group Endowed Chair in Networking
Department of Computer Science, University of Kentucky
234 James F. Hardymon Building, 301 Rose Street
Lexington, KY 40506-0495
Phone: (859) 257-3062, Fax: (859) 323-3740
E-mail: singhal@cs.uky.edu

– Dr. Grzegorz W. Wasilkowski
Professor and Director of Graduate Studies
Computer Science Department, University of Kentucky
777 Anderson Hall
Lexington, KY 40506-0046
Phone: (859) - 257 - 8029
Fax : (859) - 323 - 1971
Email: greg@cs.uky.edu

– Dr. Jun Zhang, Professor
Department of Computer Science, University of Kentucky
763F Anderson Hall, Lexington, KY 40506-0046
Phone: (859) 257-3892, Fax: (859) 323-1971
E-mail: jzhang@cs.uky.edu

– Paul Piwowarski, Lecturer
Department of Computer Science, University of Kentucky
227 Robotics Building, Lexington, KY 40506-0046
Phone: (859) 257-3678, Fax: (859) 323-1971
E-mail: paulp@cs.uky.edu

Jianchang Yang