Distributed Dynamic Fault-Tolerant Channel Allocation for Mobile Computing[*]

Ravi Prakash
Department of Computer and Information Science
The Ohio State University
Columbus, OH 43210.
e-mail: prakash@cis.ohio-state.edu

Niranjan G. Shivaratri
4 Independence Way
NEC Systems Laboratory, Inc.
Princeton, NJ 08540.
e-mail: niran@syl.nj.nec.com

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

Abstract

Efficient allocation of communication channels is critical for the performance of wireless mobile computing systems. The centralized channel allocation algorithms proposed in literature are neither robust, nor scalable. Distributed channel allocation schemes proposed in the past are complicated and require active participation of the mobile nodes. These algorithms are unable to dynamically adjust to spatial and temporal fluctuations in channel demand (load). We present a distributed dynamic channel allocation algorithm in which heavily loaded regions are assigned a large number of communication channels, while their lightly loaded neighbors are assigned fewer channels. As the spatial distribution of channel demand changes with time, the spatial distribution of allocated channels adjusts accordingly. The algorithm described in this paper requires minimal involvement of the mobile nodes, thus conserving their limited energy supply. The algorithm is proved to be deadlock free, starvation free and fair. It prevents co-channel interference and can tolerate the failure of mobile as well as static nodes without any significant degradation in service. The algorithm is scalable and imposes low computation and communication overheads, as demonstrated by simulation experiments.

Key words: channel allocation, mobile computing, cellular communication.

[*]A preliminary version of this paper [13] was presented at the 14th ACM Symposium on Principles of Distributed Computing, August, 1995.

1 Introduction

Mobile computing has found increased applications and gained importance in recent years [6]. Mobile computing makes use of cellular/wireless communication networks to provide communication among stationary and mobile hosts. In such environments, efficient allocation of wireless channels for communication sessions is of vital importance as the bandwidth alloted for cellular communication is limited.

Wireless/cellular communication networks divide the geographical area they serve into smaller regions, called cells. Each cell has a base station, also referred to as the *mobile service station (MSS)*. The mobile service stations are connected to each other by a fixed wire network. Several *mobile hosts (MH)* may be present in a cell. The MHs can move from one cell to another. This architecture, first proposed in [8] is shown in Figure 1.

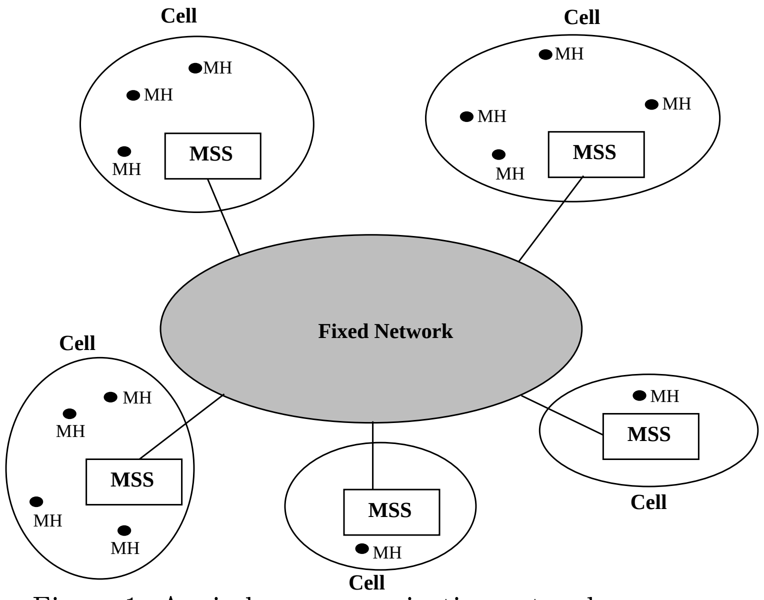

Figure 1: A wireless communication network.

To establish a communication session/place a call, an MH has to send a request to the MSS of the cell in which it is present [2]. The call can be supported if a wireless channel can be allocated for communication between the mobile host and the mobile service station. If a particular wireless channel is used concurrently by more than one call originating in a cell, or in neighboring cells, the calls will interfere with each other. Such an interference is called *co-channel interference*. However, the same wireless channel can be used to support calls in geographically separated cells such that their signals do not interfere with each other. This is known as *frequency reuse*.

The limited frequency spectrum allocated for cellular communication is divided into a finite number of wireless channels. An efficient channel allocation strategy should exploit the principle of frequency reuse to increase the availability of wireless channels to support calls. The strategy should have the following features:

1. minimize the connection set-up time
2. maximize the number of communication sessions that can be supported concurrently across the entire network
3. ability to adapt to changing load distribution in the network. The load on a cell is the rate at which new requests for establishing communication sessions originate in the cell.

To support mobile computing, the strategy should also meet the following requirements:

1. *Energy conservation:* most of the communication between mobile computers is in the form of several short bursts of data transfer. The mobile hosts have a limited energy source, in the form of a battery pack. Wireless communication drains the energy of the mobile hosts. Hence, energy should be conserved at a mobile host by keeping its involvement in the channel allocation process to a minimum. This can be achieved by minimizing the number of messages it has to exchange with the mobile service station during channel selection.
2. *Minimize hand-offs:* voice communication can tolerate hand-offs as short breaks in communication go undetected by the human ear. However, such breaks can lead to complications in data transfer to/from a mobile host. So, a channel allocation algorithm should not induce any hand-offs, over and above those caused by the movement of the mobile hosts between cells.
3. *Exploit locality of reference:* most computer applications exhibit high temporal and spatial locality of data reference. If the data items in great demand reside in a mobile host, they should be moved to a mobile service station from where they can be accessed over the fixed wire network. Until such a transfer takes place, or if such a transfer is not possible, the data references translate into frequent arrivals of requests at the mobile service station to establish communication sessions with the mobile host. A channel allocation strategy should be able to adapt to such traffic.

The channel allocation algorithms proposed in the past can be classified as

1. *Fixed Channel Allocation (FCA) strategy*: the set of channels allocated to a cell does not change with time. Mutually disjoint sets of wireless channels are assigned to neighboring cells. Each cell can use only its set of channels (the *nominal channels* of the cell) to support the calls originating from and/or directed towards the mobile hosts in its region.

2. *Dynamic Channel Allocation (DCA) strategy*: the set of channels allocated to a cell varies with time [20, 21]. A central network switch, referred to as the *Mobile Telecommunication Switching Office (MTSO)*, determines the channel(s), if any, a cell can borrow from neighboring cells, when the cell cannot support calls using its own set of channels. The central switch ensures that the borrowing does not lead to any co-channel interference.

In a *Hybrid Channel Assignment* strategy [9], the nominal channels assigned to a cell are made up of two sets: one that can only be used locally, and the other that can be borrowed by neighboring cells.

Paper Objective

In this paper we present a distributed, dynamic channel allocation algorithm. The algorithm does not need a central network switch. The mobile service station of a cell makes all the decisions about channel allocation in that cell, based on the information available locally. The MSS only needs to exchange information with its neighbors within the co-channel interference range. Unlike the FCA algorithms, the proposed algorithm can adapt to changing load distribution in the network. It is more robust than existing DCA algorithms as it does not depend on a central network switch whose failure can bring down the entire network. The algorithm also exploits the temporal locality of load distribution to make quick decisions about channel allocation. Moreover, a fast and expensive mainframe acting as the MTSO can be replaced by a set of microprocessor based switches at the $MSSs$. These switches can collectively outperform the mainframe and cost much less. The symmetry of the channel allocation procedure across the entire network makes the system scalable. The proposed algorithm meets the requirements mentioned above: it conserves energy at the mobile hosts, does not induce any hand-offs of its own, and exploits locality of reference to improve the performance. Results from a simulation study support the above assertions.

The proposed algorithm also has the following features:

1. *Bounded latency*: no mobile user that wishes to acquire a wireless channel for a communication session is made to wait indefinitely before it is either allocated a channel or is

informed of a failure to do so. Bounded latency is desirable to guarantee a certain quality of service to the users.

2. *Deadlock freedom*: there is no possibility of finding a set of mobile service stations involved in a circular wait while trying to satisfy channel allocation requests. So, the algorithm always makes progress. Resources are not wasted in detecting or resolving deadlocks.

3. *Symmetry*: All the cells follow the same procedure for channel allocation. This makes the system scalable. There is no need to design new hardware, or develop new software if more cells are to be added.

4. *Finite number of messages*: Each new request for channel allocation, to support a communication session, originating in the system leads to the exchange of a finite number of messages between the mobile service stations, before a decision to allocate a channel to it is made. Thus the fixed wire network connecting the mobile service stations is not unduly burdened.

5. *Low system overhead and network traffic*: As the proposed algorithm adapts to the locality of load distribution, each new channel allocation request is handled with an exchange of zero or a small number of messages between the mobile service stations.

6. *Concurrency*: requests for channel allocation originating independently and concurrently in different cells can be processed simultaneously.

Section 2 describes the system model. Section 3 compares the channel allocation problem with the problem of mutual exclusion in distributed systems and describes why the channel allocation problem is more complex than mutual exclusion. In Section 4, a dynamic distributed channel allocation algorithm is presented. The algorithm can adjust to changes in the temporal and spatial distribution of channel demand. In Section 5, we prove that the proposed algorithm avoids co-channel interference, is deadlock free, and has low communication overheads. The advantages of the proposed algorithm over previous centralized and distributed channel allocation algorithms are described in Section 6. Section 7 presents enhancements to the algorithm that prevent cells from being starved for channels, and also relax channel transfer constraints which will lead to a reduction in blocking probability of channel requests. Section 8 describes the simulation model and presents results of a simulation study of the algorithm's performance. In Section 9 we describe how the algorithm can withstand failure of MHs and $MSSs$. Finally, conclusions are presented in Section 10.

2 System Model and Definitions

We assume a cellular communication system that divides the geographical region served by it into hexagonal cells, with a mobile service station in the center of each cell. A mobile service station can be in wireless communication with the mobile hosts in its cell (for example, through an omni-directional antenna). A mobile host can either be a cellular telephone or a mobile computer. Calls involving cellular telephones and data transfers involving mobile computers will collectively be referred to as *communication sessions*. All the cells, except those at the boundaries of the region, have six neighbors, as shown in Figure 2.

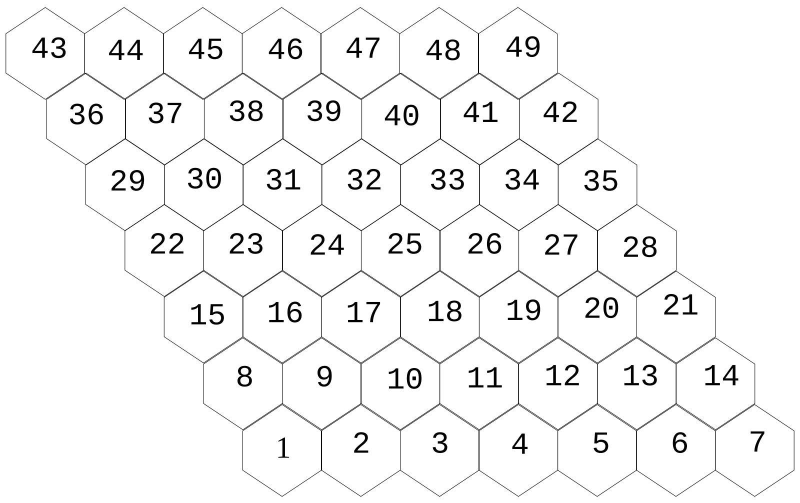

Figure 2: A 7×7 grid cellular system

The system has been assigned a frequency band that is divided into a finite number of wireless channels. These channels are independent (orthogonal) of each other. So, adjacent channel interference can be neglected. However, a channel should not be concurrently used for more than one communication session in the same cell or in neighboring cells. For example, if we assume a $3 - cell$ cluster system[1] in Figure 2, then if a channel is being used to support a communication session in cell 25, it should not be used to support another concurrent communication session in cells 17, 18, 24, 25, 26, 32 and 33. However, the channel may be used concurrently in a cell that is two hops away. Some of the wireless channels are set aside to be used exclusively for the control messages sent during link set-up between a mobile host and the mobile service station of the cell in which the mobile host is present (*control channels*). The remaining channels are

[1]For a $3 - cell$ cluster system, a channel can be used to support at most one communication session at any given time in a cluster of three mutually adjacent cells.

used to support calls (*communication channels*).

A mobile host can communicate with other units, mobile or static, only through the mobile service station of the cell in which it is present. A mobile host initiates the channel allocation protocol when it wants to establish a new communication session, or when it is informed by the mobile service station about the arrival of a communication request from some other unit. Thus, from the point of view of channel allocation, the two cases are similar. If the mobile service station determines that the connection request can be satisfied, it allocates a communication channel for the mobile host to communicate with the mobile service station for the duration of the session. From the mobile service station the signals can be forwarded along the fixed wire network, or along another wireless channel, depending on whether the other party involved in the communication session is a unit outside the cell or a mobile host in the same cell, respectively. After the session is over, the same channel can be used to support another session, either in the same cell or in neighboring cells.

For simplicity of explanation, inter-cell movement, and the resultant hand-off, can be treated as the end of the communication session in the cell from which the mobile host has moved out, and the beginning of a new communication session in the cell to which it has moved. However, to maintain continuity of service, the resultant channel allocation for hand-off should be assigned higher priority than requests for new communication sessions. Two priority schemes for hand-offs have been proposed and evaluated in [7].

3 Channel Allocation vs. Mutual Exclusion

In the context of a cell and its neighbors, the use of a particular channel to support a communication session is equivalent to a critical section execution by the cell in which the channel is being used. Several neighboring cells may be concurrently trying to choose channels to support sessions in their region. This can lead to conflicts because the number of communication channels is limited. The resolution of such conflicts is similar to the mutual exclusion problem [3, 16].

However, the channel allocation problem is more general than the mutual exclusion problem. Firstly, a cell may be supporting multiple communication sessions, from different mobile hosts, in its region, each session using a different communication channel. This is equivalent to a cell being in multiple, distinct critical sections concurrently. Secondly, existing mutual exclusion algorithms for distributed systems [3, 10, 12, 14, 16] assume that a node specifies the identity of the resource it wants to access in a critical section. Depending on the availability of that resource, appropriate decisions can be made. However, in distributed channel allocation, a cell asks for *any* channel as long as there is no co-channel interference. Due to the non-specificity of the request and because neighboring mobile service stations make channel allocation decisions independently based on locally available information, the decision process becomes more difficult.

Moreover, existing distributed mutual exclusion algorithms do not impose any upper bound on the time from the instant a node issues a request for the resource to the instant the node is granted that resource. These algorithms are not suitable for the channel allocation problem that requires the decisions to be made quickly, in real-time. So, a conservative approach that makes the channel allocation decisions quickly needs to be adopted. Such an approach may drop calls/communication requests that a more general but time consuming approach would have supported. This is a trade-off that has to be accepted.

4 A Dynamic Channel Allocation Algorithm

In the proposed algorithm, a mobile service station makes all the channel allocation decisions on behalf of the mobile hosts in its cell. Requests timestamped with Lamport's clock [10] are sent by a mobile service station to neighboring mobile service stations to determine the channel to be assigned for a communication session. Sometimes a channel needs to be deleted from a cell's set of allocated channels and transferred to another cell's set of allocated channels to support communication sessions in the latter. The distributed nature of the algorithm, and the finite but non-deterministic propagation delays of messages between mobile service stations can lead to co-channel interference if a naive channel transfer strategy is employed: multiple cells in each other's interference range may concurrently and independently decide to transfer the same channel from a mutually adjacent cell. Such a possibility is prevented as follows: having selected a communication channel for transfer, based on a round of message exchange with its neighbors, the mobile service station sends the channel identity to the neighboring mobile service stations. Only if all the neighboring mobile service stations approve of the selection is the channel transferred, otherwise not.

The set of channels allocated to a cell varies with time. Unlike existing DCA algorithms [20, 21], a newly acquired channel is not relinquished by a cell on completion of the communication session it was supporting in the cell. Instead, the channel remains allocated to the same cell until it has to be transferred to a neighboring cell. This enables the algorithm to adapt to temporal and spatial changes in load distribution. It also helps reduce the traffic due to channel allocation requests in the fixed wire network.

4.1 Lamport's Logical Clocks

In this section, we give a brief description of Lamport's logical clocks [10] as the proposed algorithm makes use of this clock system to totally order the requests for channels among neighboring cells.

Under this clock system, every process P_i maintains a clock T_i. This clock can be thought of as a function that assigns a number $T_i(s)$ to any event s at process P_i. $T_i(s)$ is referred to as the timestamp of event s. The clock/function can be implemented by a simple counter. Typically, the timestamp of an event is nothing but the value of the clock when the event occurs.

The rules for updating the clock at individual processes are as follows:

Rule 1: Clock T_i at process P_i is incremented between any two successive events at P_i as follows:

$$T_i = T_i + d \qquad (d > 0)$$

If s and r are two successive events in process P_i and event s precedes event r, then $T_i(r) = T_i(s) + d$.

Rule 2: Let s be an event of sending a message from process P_i to process P_j. Then $T_i(s)$ is obtained by applying Rule 1 and this timestamp is sent along with the message. When process P_j receives this message, the clock at process P_j is updated as follows.

$$T_j = T_j + d \qquad \text{(as per Rule 1).}$$
$$T_j = max\{T_j, T_i(s) + d\} \qquad (d > 0)$$

Now, if event r at P_j is the event of receiving the message sent by the event s at P_i, then $T_j(r) = T_j$, where T_j is obtained after performing the above two steps.

In the above rules, a value of 1 is used typically for d.

Total ordering: By using Lamport's logical clocks, the set of all events in a distributed computation can be totally ordered as follows. We denote the total ordering relation by $\Rightarrow$. If s is any event at process P_i and r is any event at process P_j, then

$$\begin{aligned} & s \Rightarrow r \text{ if and only if} \\ & \qquad T_i(s) < T_j(r) \\ & \qquad\qquad \text{OR} \\ & \qquad T_i(s) = T_j(r) \text{ and } P_i \prec P_j \end{aligned}$$

where $\prec$ is any arbitrary relation that totally orders the processes to break ties. The relation $\prec$ can be implemented in a simple way by assigning unique identification numbers to processes. Then $P_i \prec P_j$, if $i < j$.

4.2 Data Structures

All the communication channels in the system are collectively represented by a set *Spectrum*. We assume that all the channels are ordered. The channel with the lowest frequency band is considered to be the first channel and the channel with the highest frequency band is the n^{th} channel, where n is the total number of channels available.

The set of channels allocated to cell C_i is represented by $Allocate_i$. Initially, $Allocate_i$ is an empty set for every cell C_i. A subset of $Allocate_i$, known as $Busy_i$, represents the set of channels being used by C_i to support communication sessions at a particular instant of time. When a new communication request originates in C_i, one of the non-busy channels in $Allocate_i$ is assigned to support the communication session. If there is no such channel, then after a round of message exchange with the neighbors, a channel that is in the *Spectrum*, but not in the *Allocate* set of the cell or any of its neighbors is added to $Allocate_i$ as well as $Busy_i$. This channel is used to support the session. If such an attempt fails, C_i tries to transfer a non-busy channel from the *Allocate* set of its neighbors to $Allocate_i$. If such a transfer is not possible, the communication request is dropped. Otherwise, the communication is successfully completed. The set $Transfer_i$ at C_i consists of the channels earmarked for transfer from C_i to one of its neighbors. *Transfer* sets are initially empty at all the cells. T_i is the clock value maintained at cell C_i as per the Lamport's logical clock [10]. Initially, T_i is zero at every cell. RT_i is the timestamp of the current channel request. If RT_i is equal to zero, then cell C_i is not requesting a channel. Initially RT_i is equal to zero. All these data structures are maintained by the corresponding mobile service stations.

Several new communication requests may originate in a cell concurrently. These new requests, originating in the same cell, may be ordered according to a policy decided *a priori*. Only after the mobile service station has made a channel allocation decision about one locally originating request, does it process the next locally originating communication request in the sequence.

4.3 The Algorithm

(A) When a communication session is to be set-up in cell C_i, the following actions are taken by its mobile service station (MSS):

1. $T_i \leftarrow T_i + 1$;

2. $RT_i \leftarrow T_i$ /* RT_i has the timestamp of this channel request. */

3. If $Available_i \leftarrow Allocate_i - Busy_i - Transfer_i \neq \Phi$, then

   A highest order channel k from $Available_i$ is selected to set-up the session;
   $Busy_i \leftarrow Busy_i \cup \{k\}$;
   Go to step 10;

   else /* $Available_i = \Phi$ */

   Send timestamped REQUEST messages to each neighbor C_j.

4. When C_i's MSS has received REPLY messages from each of its neighbors, containing their *Allocate, Busy and Transfer* sets, it takes the union of $Allocate_i$ and the *Allocate* sets received in the REPLY messages, and stores the result in $Interfere_i$.

5. If $Free_i \leftarrow Spectrum - Interfere_i \neq \Phi$, then a channel of the highest order is selected from $Free_i$ and added to $Allocate_i$. This channel is used to support the communication session. So, it is added to $Busy_i$ as well. Then go to step 10.

6. If $Free_i = \Phi$, it does not mean that no channel is available for allocation. Perhaps, the communication session can be supported by transferring a channel. C_i's MSS takes the union of $Busy_i$, $Transfer_i$, and $Busy$ and $Transfer$ sets received in the REPLY messages in step 4, and stores the result in $Interfere_i$.

7. If $Free_i \leftarrow Spectrum - Interfere_i = \Phi$, then the communication request is dropped. Otherwise, the channel of the lowest order in $Free_i$ is chosen for the transfer.

8. Let the channel selected for transfer be k.

   $Busy_i \leftarrow Busy_i \cup \{k\}$;

   $Allocate_i \leftarrow Allocate_i \cup \{k\}$;

   C_i's MSS sends TRANSFER(k) messages to all the neighbors whose $Allocate$ sets have k as a member and waits for replies. Let S denote the set of these neighbors.

9. If all the cells in S reply AGREED:

   Channel k is used to support the communication session.

   C_i's MSS sends RELEASE(k) messages to all the cells in S.

   Go to Step 10.

   Otherwise: /* Some cells have sent REFUSE message. */

   $Allocate_i \leftarrow Allocate_i - \{k\}$;

   $Busy_i \leftarrow Busy_i - \{k\}$;

   C_i's MSS sends KEEP(k) messages to all the cells in S.

   C_i's MSS selects the next channel from $Free_i$, with order greater than that of k, and steps 8 and 9 are repeated. [2] To avoid excessive channel transfer overheads, under heavy load situations, the number of transfer attempts can be limited to the minimum of a THRESHOLD value (parameter of the algorithm) and the cardinality of $Free_i$. If all attempts to transfer a channel fail, the communication request is dropped.

10. Once a cell has decided to drop a request or to use a channel to support the corresponding communication session:

    - it sends all the deferred REPLYs to its neighbors.
    - $RT_i \leftarrow 0$;

11. When a communication session terminates in C_i, the corresponding channel is deleted from the set $Busy_i$.

(B) When a cell C_j's MSS receives a REQUEST message from C_i's MSS with timestamp T_i:

[2]The KEEP messages can be piggybacked on TRANSFER messages, if they are going to the same cell.

$T_j \leftarrow T_j + 1$;

$T_j \leftarrow \max(T_j, T_i + 1)$;

C_j's MSS sends a REPLY message to C_i if C_j is not requesting a channel (i.e. $RT_j = 0$), or if C_j is requesting a channel and C_i's request's timestamp is smaller than C_j's request's timestamp (i.e., $T_i < RT_j$ or $T_i = RT_j$ and $i < j$). Otherwise, the REPLY is deferred. As C_i only uses the union of the $Busy_j$ and $Transfer_j$ sets received in the REPLYs, in Step (A).6, and never uses the two sets separately, the communication overheads can be reduced by taking their union at C_j and sending the result, rather than both the sets, in the REPLY message. Therefore, the REPLY message contains $Allocate_j$, and the union of $Busy_j$, and $Transfer_j$.

(C) When a cell C_j's MSS receives TRANSFER(k) message from C_i:

If ($k \in Busy_j$) OR ($k \in Transfer_j$) then send REFUSE(k) message to C_i. Otherwise $Transfer_j \leftarrow Transfer_j \cup \{k\}$; Send AGREED($k$) message to C_i.

(D) When C_j's MSS receives a RELEASE(k) message, the following actions take place.

$Allocate_j \leftarrow Allocate_j - \{k\}$;

$Transfer_j \leftarrow Transfer_j - \{k\}$;

(E) When C_j's MSS receives KEEP(k) message, the following actions take place.

$Transfer_j \leftarrow Transfer_j - \{k\}$;

5 Correctness Proof

Lemma 1 *The channel allocation algorithm ensures that neighboring cells do not use the same channel concurrently.*

Proof: Let Nbr_i denote the set of neighboring cells of C_i such that concurrent use of a channel in C_i and a cell in Nbr_i will lead to co-channel interference. We have to prove the following assertion: $Busy_i \cap Busy_j = \Phi, \forall C_j \in Nbr_i$.

Initially, the assertion is trivially true as the sets are empty. Also, $Busy_i \subseteq Allocate_i$ under all circumstances. $Busy_i$ can change under three situations:

1. *In step (A).3, when* $Available_i \neq \Phi$: Let cell C_i select channel k (an element of $Allocate_i$) to support a new communication session. Assuming $Busy_i \cap Busy_j = \Phi$ and $Allocate_i \cap Allocate_j = \Phi$ prior to the addition of k to $Busy_i$, $(Busy_i \cup \{k\}) \cap Busy_j = \Phi$. So, the assertion holds after k is selected to support a call in cell C_i.

2. $Available_i = \Phi$ *in step (A).3 and* $Free_i \neq \Phi$ *in step (A).5*: Channel $k \in Spectrum$ - $(Allocate_i \bigcup_{j \in Nbr_i} Allocate_j)$ is added to $Busy_i$ and $Allocate_i$. The assertion is proved by contradiction. Let us assume that cell C_i, and its neighbor C_j, are using channel k concurrently. Cell C_j does not transfer channel k to its neighbor C_i as long as $k \in Busy_j$. This implies that the co-channel interference mentioned above can arise only if the *Allocate* sets in the REPLYs received by the mobile service stations from each other in step (A).4 did not contain k. Based on the pattern of REQUEST and REPLY messages exchanged between the two nodes, the following three situations arise:

   (a) C_i sends a REPLY to C_j before sending its own REQUEST. So, C_i's REQUEST has a higher timestamp than C_j's REQUEST. When C_j receives this REQUEST, it defers the REPLY until it has decided to use k. Then C_j sends its *Allocate* set, containing k, in the REPLY to C_i. So, C_i cannot select channel k.

   (b) C_j sends a REPLY to C_i before sending its own REQUEST. This is the similar to the previous case. So, C_i selects channel k, while C_j does not.

   (c) Both C_i and C_j receive each other's REQUEST after sending their own REQUESTs. Both the cells compare their own channel request timestamp with that received in the REQUEST message from the other. As the timestamps are fully ordered by the Lamport's clock system, the cell whose request happens to have the lower timestamp among the two requests, will defer its REPLY until it has made its own decision. The other cell will send a REPLY. Let C_i be the cell that deferred the REPLY. If C_i decides to use k, then C_j receives this information (Allocate set) in the REPLY it receives from C_i. So, C_j will not use channel k.

   Thus, two neighboring cells will not be allocated the same channel concurrently.

3. $Free_i \neq \Phi$ *in step (A).7 and AGREED messages received from all cells in S in step (A).9*: Channel $k \in Spectrum-$ $(Busy_i \bigcup Transfer_i \bigcup_{j \in Nbr_i} Busy_j \bigcup_{j \in Nbr_i} Transfer_j)$ is added to $Allocate_i$ and $Busy_i$. TRANSFER(k) is sent to the neighbors. If any neighboring cell is using k, it sends a REFUSE message. So, channel k is not used in cell C_i. From

steps (C), (D), and (E) it can be inferred that in response to a TRANSFER(k), a cell C_j sends AGREED to at most one neighbor at any time. All other TRANSFER(k) messages received by C_j, after k is added to $Transfer_j$ and before RELEASE(k) or KEEP(k) are received, are responded to with a REFUSE message. Therefore, two neighboring cells cannot simultaneously acquire channel k as a result of a transfer attempt.

∎

Lemma 2 *Each new request for a communication session originating in a cell C_i causes a finite number of messages to be exchanged between the mobile service stations of the cell and its neighbors.*

Proof: Three situations can arise. If the channel request can be satisfied locally (step (A).3), no messages are exchanged between the mobile service stations. If $Free_i \neq \Phi$ in step (A).5 at most $2N$ messages are generated to allocate a channel to the communication session, where N is the number of neighboring cells in the co-channel interference range: N REQUESTs from C_i to its neighbors, and a REPLY from each neighbor to C_i. If $Free_i = \Phi$ in step (A).7, the request is dropped after the exchange of the same $2N$ messages. Finally, if $Free_i \neq \Phi$ in step (A).7, $5N \leq$ *messages needed to make a channel allocation decision* $\leq 2N + 3N \times minimum(\mid Free_i \mid, THRESHOLD)$. Besides the $2N$ messages already mentioned, at most N TRANSFER(k) messages from C_i to its neighbors, an AGREED or REFUSE message from each neighbor to C_i, and finally a RELEASE(k) or KEEP(k) to each neighbor are needed per channel transfer attempt. The number of attempts is upper bound by the minimum of $\mid Free_i \mid$ and THRESHOLD. As THRESHOLD is a constant value chosen as a parameter of the algorithm, the message complexity is $O(N)$. ∎

Lemma 3 *The channel allocation algorithm is deadlock free.*

Proof: New channel requests originating concurrently in different cells get totally ordered by their timestamps. A mobile service station with REPLYs pending to its own REQUESTs, sends REPLYs to all REQUESTs with a lower timestamp and defers other REPLYs. As the same ordering of channel requests is seen by all the nodes, there is no circular deferring of REPLYs among the mobile service stations.

During the interval between sending a TRANSFER(k) message to the neighbors, and receiving either a REFUSE or an AGREED message from each neighbor, a cell does not suspend replying to TRANSFER(k) messages it may itself receive from the neighbors. Instead, it responds to such transfer attempts with a REFUSE message during this interval. This conservative policy may lead to some requests, that could have otherwise been supported, being dropped. However, it avoids any circular wait during the channel transfer attempts, thus preventing deadlocks. ∎

6 Comparison with Earlier Work

The proposed algorithm has several advantages over existing channel allocation algorithms. In the centralized algorithms, the central network switch is the single point of failure that can bring down the entire network, and can become a bottleneck during high load situations. The proposed algorithm is more robust and does not have a bottleneck as the inter-MSS message traffic is distributed over the entire network. Each mobile service station shoulders responsibility. The size of the messages is also small because very little information is exchanged.

The algorithm adapts well to changing load distribution. Due to statistical fluctuations, there may be temporally and spatially distributed pockets of high load in the system. High load situations may also arise due to the locality of reference to data items residing on mobile hosts present in those hot-spots. The channel transfer feature of the proposed algorithm ensures that unused channels are moved from lightly loaded cells to the heavily loaded cells. Therefore, most of the channel requests that originate in heavily loaded cells can be satisfied locally by selecting a free channel from the *Available* sets. Moreover, if a mobile host, containing frequently accessed data, moves from a cell to a neighboring cell, channels are transferred from the *Allocate* set of the former to the latter, over a period of time. Thus the size of the *Allocate* sets of cells can adapt with time to support the locality of data reference.

The algorithm has low computational overheads. Most of the steps involve union, intersection or subtraction of sets of channels, which can be efficiently carried as operations on bit-streams, with a bit for each channel. As already mentioned, the hardware cost can be reduced, while maintaining the performance, by replacing an expensive mainframe, acting as the central network switch, with inexpensive microprocessor based controllers at the mobile service stations.

In [4], channel allocation is done by the mobile host and the mobile service station working together. In mobile computing, most of the communication is in the form of several short bursts of unidirectional data transfer, with unpredictable interval between two successive bursts. If the mobile host had to expend energy in channel allocation each time such a transmission is needed, it would soon become a significant overhead. In the proposed algorithm, the involvement of the mobile host in channel selection is limited to sending a request to its mobile service station for uplink connectivity and receiving a message from the mobile service station carrying the identity of the selected channel, if any. This leads to significant energy savings at the mobile host.

Channel allocation strategies that constantly monitor the *signal-to-interference ratio* of channels, and employ cell sectoring [17] or cell overlaying [11] have been proposed. However, these strategies are complex to implement and have a higher probability of needing hand-offs [19]. In addition to inter-cell hand-offs, intra-cell hand-offs may be needed. For example, when cell sectoring is employed, a cell is divided into multiple sectors. Some channels can be used to support communication sessions in only particular sectors of a cell. If a mobile host continues to use the same channel as it moves from one sector to another, in the same cell, co-channel interference may result. Hence, an intra-cell hand-off may be required. In the case of cell overlaying, the regular hexagonal grid of cells is overlaid with smaller cells. Channels used in the overlaid cells can be used at smaller distances. However, if a mobile host using such a channel moves out of the smaller overlaid cell while remaining in the bigger underlaid cell, co-channel interference may occur. Once again hand-offs may be required to avoid interference. Increased rate of hand-offs will lead to a significant degradation in the quality of service for data transmission to and from mobile hosts for two reasons. First, hand-offs impose computational overheads on the MHs and $MSSs$ in the system. Second, data communication has a much lower tolerance for hand-off induced temporary breaks in communication than voice communication. The proposed algorithm does not induce any intra-cell hand-offs. Hence, it does not suffer from the two problems mentioned above.

The distributed nature of the proposed algorithm makes the cellular network scalable. Channel allocation decisions are made by each mobile service station locally. All the messages needed to set up a communication session in a cell are restricted to that cell and its immediate neighbors. So, the traffic on the wired network between adjacent mobile service stations does not increase with increasing number of cells; it only increases with increasing load in the cell and its neighbors. For the centralized algorithms, the traffic on the communication paths leading to the central network switch increases with increasing network size. So, with the centralized algorithm, as the network expands, existing links will have to be replaced with those with a higher bandwidth.

7 Enhancements

While the algorithm is deadlock free, there is a possibility of some cells being starved for channels. For example, all the channels of a cell may be transferred to its neighbors. Later when the cell needs a channel, the neighbors may happen to be so highly loaded that the cell cannot acquire any channel from them.

Sometimes channel transfer attempts may be denied even if the transfer would not have caused co-channel interference. For example, let cell C_j add channel k to $Transfer_j$ on receiving a TRANSFER(k) message from C_i. Later, when another neighbor C_l of C_j tries to transfer channel k from C_j, it will be sent a REFUSE(k) message even if C_l and C_i are not mutually neighboring cells and cannot cause co-channel interference in each other's region.

Channel starvation in cells, and unavailability of channels due to their presence in *Transfer* set can be mitigated by making the following enhancements to the algorithm:

1. In cell C_i, instead of maintaining the set $Transfer_i$ for all the channels earmarked for transfer from C_i to its neighbors, the following sets are maintained: $\forall j : C_j \in Nbr_i$, $Transfer_{ij}$ is maintained. $Transfer_{ij}$ consists of the channels earmarked for transfer from C_i to C_j or to a cell that is a neighbor of C_i as well as C_j. Then $Transfer_i = \cup_{j \in Nbr_i} Transfer_{ij}$.

2. In step (B) of the algorithm, on receiving a REQUEST from C_i, C_j sends $Transfer_{ji}$ in the REPLY instead of $Transfer_j$.

3. In steps (D) and (E) of the algorithm, on receiving a RELEASE(k) or KEEP(k) message from C_i, C_j deletes the channel k from $Transfer_{ji}$ and $Transfer_{jl}$ for all C_l such that C_l is a neighbor of C_i as well as C_j.

4. Step (C) of the algorithm can be rewritten as:

   If $(k \in Busy_j)$ OR $(k \in Transfer_{ji})$ OR $(\mid Allocate_j \mid < LOW\_THRESHOLD)$ then send REFUSE(k) message to C_i. Otherwise $Transfer_{ji} \leftarrow Transfer_{ji} \cup \{k\}$, $Transfer_{jl} \leftarrow Transfer_{jl} \cup \{k\}$ for all C_l such that C_l is a neighbor of C_i as well as C_j; Send AGREED(k) message to C_i.

LOW_THRESHOLD is a constant value introduced as a parameter of the algorithm. A cell will agree to transfer a channel to its neighbor only if it has at least LOW_THRESHOLD number of channels. Thus, channel starvation in cells is avoided. The degree of dynamism of the algorithm can be varied by changing the value of LOW_THRESHOLD.

Maintaining a transfer set with respect to each neighbor avoids the above mentioned problem of transfer requests being denied even if such transfers would not lead to co-channel interference.

The use of logical clocks to timestamp channel requests ensures fairness. If cell C_i's channel request causally precedes a neighboring cell C_j's channel request, C_i's request is processed before C_j's request. Thus, with the enhancements mentioned above, the algorithm ensures deadlock freedom, avoids starvation, and guarantees fairness.

A communication session of a mobile host may be disrupted if the mobile host moves from one cell to a neighboring *target* cell and no channel is available in the target cell to support the session. Such a disruption is referred to as hand-off failure.

Two priority based hand-off strategies have been proposed in [7]. These strategies can be easily incorporated in the algorithm described in Section 4. The two strategies are:

1. **Channel Reservation:** a certain number of channels (H) are reserved exclusively for hand-offs. Any available channel in the target cell can be used to support handed-off communication sessions. However, if a channel is needed for a new communication session arising in the cell and the number of available channels is less than H, the channel request is denied.

2. **Hand-off Queue:** channel requests for hand-offs are queued up on an FCFS basis in the target cell. If a channel becomes available in the target cell, and its hand-off queue is non-empty, the channel is used to satisfy the hand-off request at the head of the queue. An upper bound is imposed on the length of the queue and the duration for which a hand-off request can be queued up. This is to ensure that hand-offs, if possible, are done fast enough so that the disruption in the communication session during hand-off is not noticeable.

Simulation results in [7] show that both the strategies reduce the hand-off failure probability significantly. However, the first strategy leads to a corresponding increase in the probability of new channel requests being dropped. Hence, the hand-off queuing strategy appears to be the better of the two strategies.

Disruptions in the performance of the algorithm, due to the failure of some $MSSs$, can be handled by replicating the channel allocation information in an MSS at a few other $MSSs$ as described in [1]. Either optimistic or pessimistic information replication can be done.

8 Simulation Experiments and Results

In Section 2 we described the system model for a mobile computing environment and in Section 4 we presented a distributed dynamic channel allocation algorithm for such an environment. As already explained, the algorithm is deadlock free, does not have a single point of failure, and can easily adapt to changes in demand for communication channels across the network. The message complexity of the algorithm is low and it is energy efficient.

However, in addition to the properties mentioned above, a good channel allocation algorithm should have a good performance in situations of low demand for channels and should deny as few channel requests as possible when there is a high demand for channels. Its performance should degrade gracefully as channel demand increases and it should be scalable. In order to quantitatively measure the performance of the proposed algorithm, we conducted several simulation experiments. A process-oriented, discrete-event simulation using CSIM [15] was developed for this purpose.

8.1 Simulation Environment

Three important aspects of the simulation environment are the *channel reuse pattern*, *communication and computation characteristics*, and *channel demand distribution*.

8.1.1 Channel Reuse Pattern

We assumed a 7×7 grid of hexagonal cells, as shown in Figure 2. The cells along the border have 2, 3 or 4 neighbors while all the internal cells have six neighbors. These 49 cells form a *3-cell cluster system*. A 3-cell cluster system has the following properties: if a channel is being used to support a communication session in a cell, it cannot be used concurrently in the immediate neighboring cells [18]. However, the same channel can be used at a distance of two cells. For example, in Figure 2, a channel can be used concurrently in cells 24 and 26, but not concurrently in cells 24 and 25 or 25 and 26. So, in a cluster of three mutually adjacent cells, a channel can be used to support at most one communication session at any given time. Hence, the name *3-cell cluster*.

Besides the wireless communication channels, we assume the presence of a sufficient number of control channels in a system. Whenever an MH wants to acquire a channel for communication with the MSS of its cell, a control channel is available for sending the connection request from the MH to the MSS and for receiving the outcome of the connection request from the MSS

to the MH. The control channels are assumed to have a bandwidth of only 8 kilobits/second which is much lower than the bandwidth of the communication channels. Hence, a large number of control channels can be present in the system without taking up a significant portion of the spectrum allocated for mobile computing.

8.1.2 Communication and Computation Characteristics

We assumed that every pair of adjacent $MSSs$ were connected by a point-to-point communication link having a bandwidth of 1 megabits/second. This bandwidth is comparable to the sustainable bandwidth of Ethernet. Hence, a network of co-axial cables between adjacent $MSSs$ is sufficient for our needs. Such networks are already in place in most urban areas. As mentioned above, the wireless control channels between an MH and an MSS have a bandwidth of 8 kilobits/second. The size of each control message of the algorithm (REQUEST, REPLY, etc.) is equal to $224 + 2\times$ *number of communication channels* bits. The 224 bits include a 32 bit node-id and a 64 bit timestamp. In addition to this, a message from C_i to C_j has two bits corresponding to each channel in the *Spectrum*: one bit indicating whether the channel is currently allocated to C_i, and the other bit indicating whether the channel is currently being used in C_i to support a communication session, or marked for possible transfer from C_i (bit-wise union of $Busy$ and $Transfer$ sets).

When a message is sent by a node (MH or MSS), a constant time overhead of 2 millisecond is incurred. This includes the time to compose the message (all the bitwise boolean functions on the channel fields of the allocate and busy/transfer channel fields of the message) and the time to place it in the send buffer. Similarly, the overheads incurred on receiving a message are equal to 2 millisecond. This time includes the time to retrieve the message from the receive buffer, decode the type of the message (REQUEST, REPLY, etc.), and to execute the appropriate bitwise boolean operations on the channel fields depending on the type of the message.

8.1.3 Channel Demand Distribution

As in previous studies [20, 21], we assumed that each cell observes the same traffic pattern. The arrival rate of channel requests in each cell has a Poisson distribution with mean λ and the duration of a communication session during which a communication channel is in use is exponentially distributed with mean $1/\mu$.

We assume that $\lambda_i = \lambda_j$ and $\mu_i = \mu_j$ for all cells C_i and C_j in the system. Hence, given that the total number of communication channels in the *Spectrum* is N, mean number of channels allocated to each cell in the 3-cell cluster system is $N/3$. According to Little's result, the average number of channels needed to support all channel requests in a cell is equal to $\lambda \times (1/\mu + \textit{channel allocation time})$. Assuming that the channel allocation time for the algorithm is much less than $1/\mu$, we express *channel request load* for a cell as $\frac{3\times\lambda}{N\times\mu} \times 100$, which is an approximation of the percentage of channels allocated to a cell that are needed to support all concurrent channel requests in that cell. For example, for a 300 channel system, if the arrival rate of channels requests is 10 requests/cell/minute and the mean duration of a communication session is 3 minutes the channel request load is equal to $\frac{3\times10\times3}{300} \times 100 = 30\%$. For a 100 channel system, the same arrival rate corresponds to a 90% channel request load.

When a cell C_i fails in its attempt to transfer a channel k from its neighbors, it selects another channel k' from $Free_i$ and tries to transfer it. The number of such transfer attempts in our channel allocation scheme is less than or equal to THRESHOLD (a parameter of the algorithm). In our simulations we set THRESHOLD equal to 1. So, if the first transfer attempt fails the channel request is dropped.

8.2 Simulation Results

We assumed the average duration of communication sessions ($1/\mu$) to be 3 minutes, identical to that used in [5, 20, 21]. Simulation experiments were done for three different values of N, the number of communication channels in the *Spectrum*: 50, 100, and 400. The performance of the algorithm was evaluated when each cell C_i maintained a single transfer set $Transfer_i$ with respect to all its neighbors (henceforth referred to as *common transfer set*) as well as when each cell maintained a separate transfer set $Transfer_{ij}$ with respect to each neighbor C_j (henceforth referred to as *directional transfer set*) as described in Section 7. The arrival rate of channel requests was varied so as to vary the channel request load. Various performance metrics were measured for different channel request load situations.

In each simulation run, data was collected only after the first *fifty thousand* channel requests were processed. This was done to eliminate the impact of startup transients. Then data was collected until the next *one hundred and fifty thousand* channel requests had been processed. The mean value of ten such runs, each with a different random number seed, corresponds to a single data point in the following figures.

8.2.1 Impact of Channel Demand on Direct Connections

As described earlier, if a channel request arrives at an MSS and there are channels allocated to the cell that are neither busy, nor in the transfer set, then they can be *directly* assigned to the request, based solely on local information without having to communicate with the $MSSs$ of neighboring cells.

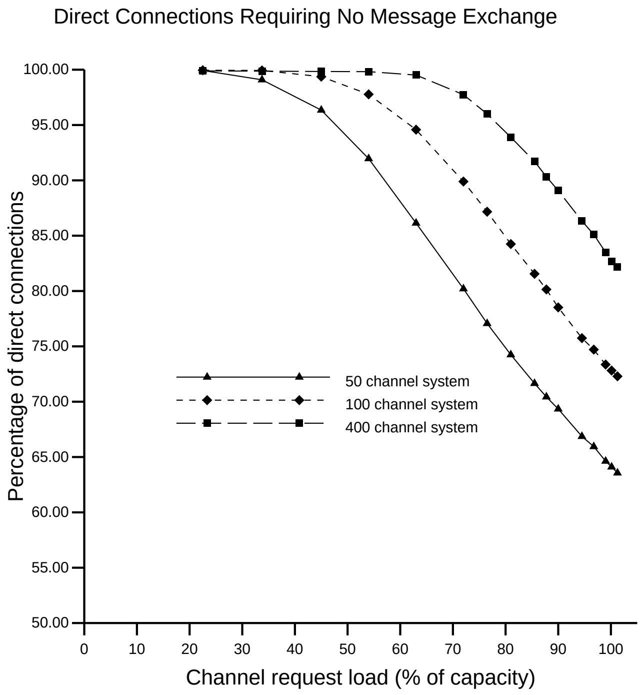

Figure 3: Variation in the percentage of channel requests satisfied with direct connection for changing channel request load.

Figure 3 shows the percentage of channel requests that lead to *direct* connections without any need for inter-MSS communication when *directional transfer sets* are used. The percentage of direct connections declines with increasing channel request load. This reduction occurs because as load increases, the probability of finding an idle channel in the cell decreases. However, two interesting properties of the algorithm are manifested in the figure.

1. Even for high channel load (close to 100%), more than 60% of the channel requests lead to direct connections without any message exchange in a system with a small *Spectrum* (50 channels).

2. For the same channel request load, as the number of channels in the *Spectrum* increases, the percentage of channel requests that lead to direct connections also increases. For example, when channel load is close to 80% the percentage of channel requests that lead to direct connections for 50, 100, and 400 channel systems are approximately 74%, 84%, and

93%, respectively. The effectiveness of the proposed channel allocation scheme becomes evident when it is noted that an 80% channel demand load for a 400 channel system corresponds to eight times as many channel requests per unit time as an 80% channel demand load for a 50 channel system.

In low load situations, most of the channel requests can be satisfied directly without any need for communication over the wireline network. As the number of communication channels in the *Spectrum* increases, the *MSSs* have greater flexibility in assigning channels and a greater proportion of channel requests can be satisfied locally. Thus the fixed wireline network is not burdened with increase in *Spectrum* size. The simulation results demonstrate that the effectiveness of the algorithm increases with increase in the size of the *Spectrum*, even when the load at cells increases at the same rate as the increase in the size of the *Spectrum*. This shows that the algorithm is highly scalable.

When *common transfer sets* are employed, the performance is comparable to the performance of the algorithm with *directional transfer sets*. Hence, the performance graph for *common transfer sets* is not shown.

8.2.2 Impact of Channel Demand on Wireline Network Traffic

The average number of inter-*MSS* messages needed for allocating a communication channel for each channel request was measured. The results are shown in Figure 4 for the simulations in which *directional transfer sets* were employed.

For low load situations almost no such messages are needed, while their number increases with increasing load. The number of inter-*MSS* messages per channel request decreases with increasing number of communication channels in the *Spectrum* for the same channel request load. However, even for channel request load close to 100%, a 50 channel system needs less than 5.5 inter-*MSS* messages per channel request on an average. The trends in Figure 4 are consistent with the results of Section 8.2.1.

It is to be noted that the size of the messages in the algorithm increases with the number of communication channels in the *Spectrum*. However, the number of messages needed per channel request decreases with increase in the number of communication channels in the *Spectrum*, for the same load. As a result, traffic in the wireline network per channel request shows little change regardless of the total number of channels in the *Spectrum*.

With load remaining the same, an increase in the number of *MSSs* will not change the number of inter-*MSS* messages per channel request, and the traffic on each inter-*MSS* link

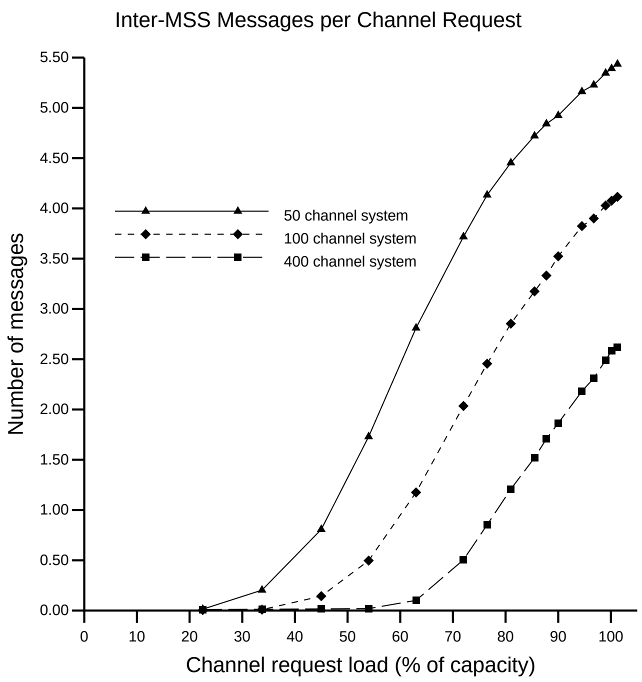

Figure 4: Variation in the number of inter-MSS control messages with changing load.

will remain the same. This is because an MSS only needs to communicate with its neighbors, regardless of the total number of cells in the system. On the other hand, in a centralized channel allocation scheme, the network traffic on the link(s) incident on the $MTSO$ increases linearly with an increase in the number of cells. Thus, the links incident on the $MTSO$, and the $MTSO$ itself, can become bottlenecks. The non-dependence of link traffic on network size, in the proposed algorithm, makes the algorithm scalable.

Once again, using *common transfer sets* leads to a performance that is comparable to the performance of the *directional transfer set* case.

The simulation results demonstrate that the volume of traffic in the wireline network connecting $MSSs$ is quite small even at high loads.

8.2.3 Impact of Channel Demand on Channel Transfer Attempts

We measured the percentage of channel requests that result in the MSS of the cell trying to transfer a channel from neighboring cells. The observed values for the *directional transfer set* case are shown in Figure 5.

As expected, there is no need for inter-cell channel transfer at low load. This is because at low loads, each cell has a sufficient number of communication channels allocated to it to serve all channel requests arising in its region. However, with increasing load, the percentage of channel

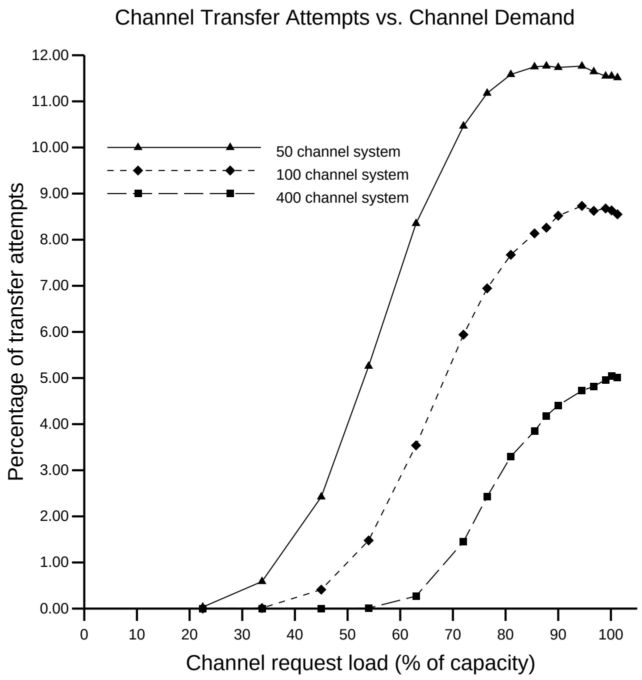

Figure 5: Variation in the percentage of channel transfer attempts with changing load.

transfer attempts increases. Still, we can see from Figure 5 that the percentage of such attempts is quite small. For a 50 channel system, less than 12% of the channel requests result in attempts to transfer channels, even for channel load as high as 100%. The small percentage of transfer attempts shows that the proposed scheme adapts quite well to changing load patterns in a group of neighboring cell.

Moreover, as the number of channels in the *Spectrum* increases, the percentage of transfer attempts goes down for the same channel load. This reduction in the percentage of transfer attempts is because when there are a large number of channels in the system, the probability of finding an idle channel locally is also higher.

When *common transfer sets* are employed, the percentage of channel requests that result in transfer attempts is comparable to the percentage for *directional transfer sets*. Hence, using *directional transfer sets* once again seems to have little impact on performance.

As the percentage of transfer attempts is so low, a change in the transfer attempt THRESHOLD from 1 to a higher value will lead to only a small increase in the number of channel requests that are satisfied, and a small decrease in the number of channel requests that are dropped. However, such an increase in THRESHOLD will increase the number of inter-MSS messages. This negligible improvement in performance may not be worth the increase in network traffic.

The simulation results discussed so far demonstrate that the proposed algorithm adapts quite well to changing load conditions. Due to its adaptive nature, most channel requests are satisfied locally and the network traffic between $MSSs$ is quite low.

8.2.4 Impact of Channel Demand on Dropped Channel Requests

We measured the percentage of channel requests that are dropped/denied because the algorithm cannot allocate any channel in response to such requests. Figure 6 shows the observed values when *directional transfer sets* are employed.

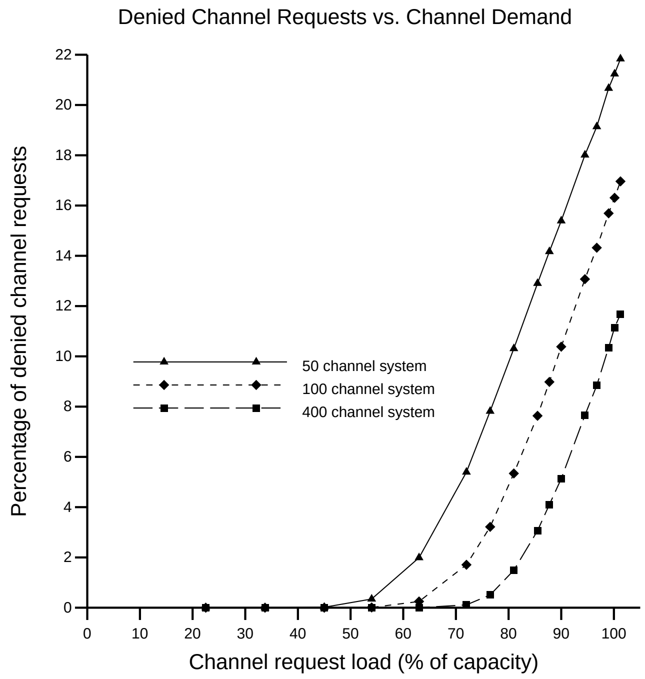

Figure 6: Variation in the percentage of dropped channel requests with changing load.

As expected, when the channel request load is low, hardly any channel request is denied. With increasing load, the percentage of requests that are denied increases. Moreover, for the same channel request load, having a larger number of channels in the *Spectrum* leads to fewer channel requests being dropped. Therefore, the algorithm scales very well with the size of the *Spectrum*.

The algorithm shows a very good performance even under very high load situations. For a 50 channel system when the load is close to 100%, less than 22% of the requests are dropped. When there are 400 channels in the *Spectrum*, less than 12% of the requests are dropped for a 100% load. This is an extremely low percentage of dropped requests for a distributed algorithm considering the fact that each MSS makes a conservative decision about channel allocation based only on the status information of its cell and its immediate neighbors. Some of the channels that were in use in the neighbors when they sent their status in the REPLY or REFUSE messages may have become available when the channel allocation decision was made. But the MSS is unaware of their availability.

It is to be noted that the percentage of channel requests that were dropped is greater than the percentage of channel requests that led to channel transfer attempts. This is because some channel requests are dropped without making a transfer attempt as described in Step (A). 7 of the algorithm.

Once again, using *common transfer sets* led to a performance comparable with using *directional transfer sets.*

8.2.5 Duration of Non-availability of Channels in Transfer Set

The simulation results described so far seem to indicate that using *directional transfer sets* has no positive impact on the performance of the channel allocation algorithm. However, the arguments presented in Section 7 seem to suggest that using *directional transfer sets* will lead to an increase in the availability of channels that would be otherwise locked up unproductively in *common transfer sets.*

In order to investigate the inconsistency between the expected and observed effect of using *directional transfer sets*, we measured the average and maximum duration that any channel is *locked up* in a transfer state and is therefore unavailable for supporting a communication session in a cell and its immediate neighbors. The maximum duration for which a channel is present in a transfer set, when *common transfer sets* are employed, is shown in Figure 7.

It is to be noted that even in very high load situations, the maximum duration for which a channel is locked up in the transfer state is less than 0.035 seconds. The average duration for which a channel is in the transfer state was observed to be approximately 0.022 seconds. This is a very small period of time compared to the average duration of a communication session which is 3 minutes. Hence, the time for which a channel is unavailable is extremely small. The increase in the availability of channels, for this small period of time (0.035 seconds at the most), brought about by using *directional transfer sets* has an imperceptible impact on the performance of the algorithm. For example, in a 400 channel system, a 100% channel request load situation corresponds to one new channel request arriving at an MSS every 1.33 seconds. Only a small percentage of these requests will result in transfer attempts. Hence, the number of occasions when such a reduction in channel non-availability is going to be useful is also quite small.

Also, if a cell C_i identifies a channel k for possible transfer to cell C_j, not all the neighbors of C_i may have channel k in their *Allocate* sets. So, using *directional transfer sets* will increase

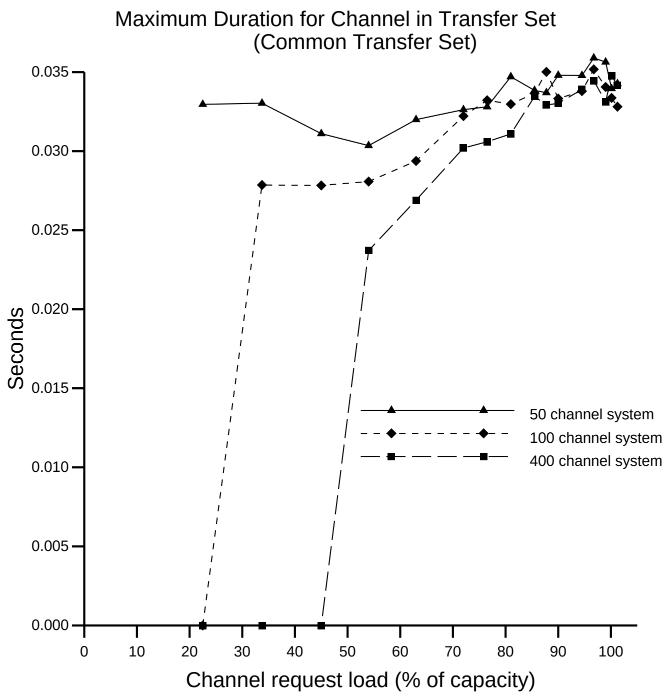

Figure 7: Variation in the time for which a channel is earmarked for potential transfer with changing load.

the availability of k only in those neighbors of C_i that have k in their *Allocate* sets and are also not immediate neighbors of C_j. For other neighbors of C_i, using *common* or directional transfer sets makes no difference as far as the availability of channel k is concerned.

Therefore, *directional transfer sets* are not very useful. The performance gain may not be worth the additional computation and memory overheads incurred at each MSS for maintaining *directional transfer sets*. Hence, *common transfer sets* are preferable due to their simplicity of implementation.

The maximum delay in making a channel allocation decision (time elapsed between an MH sending a request for a channel and being informed of the outcome of its channel allocation request) was approximately 1.2 seconds (for a 400 channel system at close to 100% load), representing the worst case behavior of the algorithm. Typically, channel transfer attempts lead to the greatest delays in making channel allocation decisions. The mean delay in making a channel allocation decision, when channel transfer was attempted, was approximately 0.3 seconds. This delay is acceptable considering that we assumed modest bandwidths for the wireless and wireline communication links.

9 Fault Tolerance

In a mobile computing environment, both the MHs and $MSSs$ are prone to failure. The failure probability of MHs is much higher than that of the $MSSs$ as the former are exposed to hostile conditions.

When an MH fails in the middle of a communication session, the session is terminated. Hence, the channel that was being used for the communication session is no longer in use. The corresponding MSS detects the failure of the MH in its cell and deletes the channel from its $Busy$ set. Thus, as far as channel allocation is concerned, failure of an MH is conceptually as simple to handle as the completion of a communication session.

In the context of the proposed channel allocation algorithm, handling the failure of an MSS is a little more involved. We assume that MSS failures are fail-stop in nature. When MSS_i of cell C_i fails, all the communication sessions between MSS_i and the MHs in its cell are terminated. Hence, no channel is in use in C_i during the duration of MSS_i's failure. However, if the mobile support station MSS_j of a neighboring cell C_j sends a message for the channel allocation algorithm to MSS_i and is expecting a response, then MSS_j may have to wait indefinitely. This will jeopardize the performance of the channel allocation algorithm.

The indefinite wait of MSS_j due to the failure of MSS_i can be handled provided the following holds for the wireline network connecting the $MSSs$:

1. The underlying protocol at lower layers of the network can distinguish between MSS failure and link failure. So, failure of a link between neighboring nodes MSS_i and MSS_j is not misinterpreted by either MSS as the failure of the other MSS.

2. In the event of link failure, there exist alternate paths in the underlying wireline network for communication between physically adjacent $MSSs$ through another $MSSs$ that is a common neighbor of both the $MSSs$.

Therefore, if the link between neighboring mobile service stations MSS_i and MSS_j fails, messages between them can be routed using the MSS_i-MSS_k and MSS_j-MSS_k links, where MSS_k is adjacent to both MSS_i and MSS_j in the wireline network. In the worst case, if all the wireline links connecting MSS_i to its neighboring $MSSs$ fail, MSS_i can still communicate with its neighbors, although at a lower bandwidth, using the 8 kilobits/second wireless control channels.

If MSS_j is waiting for a REPLY after sending a REQUEST message to MSS_i, it times out after a predetermined amount of time has elapsed and the REPLY has not been received. At this point of time, MSS_j assumes that: (i) MSS_i has failed, (ii) MSS_j has received a REPLY from MSS_i such that $Allocate_i$, $Busy_i$ and $Transfer_i$ received in the REPLY are empty sets. The timeout period is long enough so that MSS_j does not time out when the REPLY is delayed due to heavy load at MSS_i or due to the slowness of the link(s) between MSS_i and MSS_j. When MSS_j times out waiting on MSS_i, MSS_j deletes MSS_i from its list of neighbors. In the future, channel allocation messages are not sent to MSS_i. If MSS_j times out on MSS_i after sending a TRANSFER(k) message, MSS_j proceeds with the channel allocation algorithm assuming that it has received an AGREED(k) message from MSS_i.

Later, when MSS_i is repaired, it sends a notification message to the $MSSs$ of all the neighboring cells. When MSS_j receives the notification message, it once again adds MSS_i to its list of neighbors. At this point of time if MSS_j is waiting for responses after sending a REQUEST or TRANSFER message to its neighbors, it sends the same message to MSS_i. MSS_i can resynchronize its local clock using the timestamp of the first message corresponding to the channel allocation algorithm it receives after its recovery. Thus, a repaired MSS is fully integrated with the rest of the network.

10 Conclusion

An efficient channel allocation algorithm is important for high utilization of a cellular communication network. In this paper, we presented an efficient distributed dynamic channel allocation algorithm. The algorithm distributes the responsibility for channel allocation among the mobile service stations of the network. This is a departure from the algorithms proposed in the past, which employed a centralized controller.

The algorithm is especially suited for supporting mobile computing. It keeps the involvement of mobile hosts in channel selection to a minimum, thereby conserving the limited energy at their disposal. The algorithm keeps the number of hand-offs to a minimum as it does not induce any intra-cell hand-offs, unlike some strategies proposed in the past. It also exploits the locality of reference. The algorithm is dynamic and easily adapts to changes in the network load distribution by transferring allocated channels from lightly loaded cells to highly loaded cells.

The algorithm is deadlock free, fair and has low computational and communication overheads. The proposed enhancements prevent channel starvation in the cells. The cost of new hardware needed to implement the algorithm is low. The distributed nature of the algorithm and the symmetry of the channel allocation procedure across the entire network makes the system scalable. The simplicity of the algorithm makes it easy to implement on an actual network.

The algorithm can easily withstand failures of MHs and $MSSs$ without significant degradation in performance.

Results of simulation experiments indicate that most of the channel requests from the MHs in a cell can be satisfied locally by directly assigning a channel in the cell's *Allocate* set. This is true even under high load situations. As a result an extremely small number of messages need to be sent along the wireline network between adjacent $MSSs$. Only a small fraction of channel requests lead to transfer attempts. Even when the channel demand is high, a very small fraction of channel requests are dropped. An increase in the number of communication channels in the *Spectrum* leads to an increase in the number of direct connections and a decrease in the number of inter-MSS messages, percentage of transfer attempts and percentage of dropped requests, even when the arrival rate of channel requests is increased correspondingly.

Moreover, using a low overhead and simple to implement *common transfer set* leads to as good a performance as the more complicated *directional transfer sets*. This leads us to conclude that the proposed algorithm, even though simple, has performance close to the optimal performance. Attempts to further improve its performance using complicated data-structures/strategies will lead to minimal gains while incurring heavy costs in terms of memory requirements and computation power.

References

[1] S. Alagar, R. Rajagopalan, and S. Venkatesan. Tolerating Mobile Support Station Failure. Technical Report UTDCS-16-93, University of Texas, Dallas, 1993.

[2] B. R. Badrinath, A. Acharya, and T. Imielinski. Structuring Distributed Algorithms for Mobile Hosts. In *Proceedings of the* 14th *International Conference on Distributed Computing Systems*, June 1994.

[3] K. M. Chandy and J. Misra. The Drinking Philosophers Problem. *ACM Transactions on Programming Languages and Systems*, 6(4):632–646, October 1984.

[4] 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.

[5] D. D. Dimitrijevic and J. Vucetic. Design and Performance Analysis of the Algorithms for Channel Allocation in Cellular Networks. *IEEE Transactions on Vehicular Technology*, 42(4):526–534, November 1993.

[6] G. H. Forman and J. Zahorjan. The Challenges of Mobile Computing. *IEEE Computer*, 27(4):38–47, April 1994.

[7] 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 *Proceedings of the* 41st *Vehicular Technology Conference*, pages 549–553. IEEE, 1991.

[8] 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.

[9] T. J. Kahwa and N. D. Georganas. A Hybrid Channel Assignment Scheme in Large Scale, Cellular Structured Mobile Communication Systems. *IEEE Transactions on Communication*, 26(4), April 1978.

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

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

[12] M. Maekawa. A $\sqrt{N}$ Algorithm for Mutual Exclusion in Decentralized Systems. *ACM Transactions on Computer Systems*, pages 145–159, May 1985.

[13] R. Prakash, N. Shivaratri, and M. Singhal. Distributed Dynamic Channel Allocation for Mobile Computing. In *Proceedings of the* 14^{th} *ACM Symposium on Principles of Distributed Computing*, pages 47–56, Ottawa, Canada, August 1995.

[14] G. Ricart and A. K. Agrawala. An Optimal Algorithm for Mutual Exclusion in Computer Networks. *Communications of the ACM*, 24(1):9–17, January 1981.

[15] H. Schwetman. *CSIM Revision 16.* Microelectronics and Computer Technology Corporation, 3500 West Balcones Center Drive, Austin TX 78759, U.S.A., June 1992.

[16] M. Singhal. A Dynamic Information-Structure Mutual Exclusion Algorithm for Distributed Systems. *IEEE Transactions on Parallel and Distributed Systems*, 3(1):121–125, January 1992.

[17] V. H. MacDonald. The Cellular Concept. *Bell System Technical Journal*, 58(1), January 1979.

[18] W. Yue. Analytical Methods to Calculate the Performance of a Cellular Mobile Radio Communication System with Hybrid Channel Assignment. *IEEE Transactions on Vehicular Technology*, 40(2):453–460, May 1991.

[19] T.-S. P. Yum and W.-S. Wong. Hot-Spot Traffic Relief in Cellular Systems. *IEEE Journal on Selected Areas in Communications*, 11(6):934–940, August 1993.

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

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