# An Efficient Distributed Channel Allocation Algorithm Based on Dynamic Channel Boundaries

Chi Chung Lam
Department of Computer and Information Science
The Ohio State University
Columbus, OH 43210
clam@cis.ohio-state.edu

## Abstract

*In a mobile computing environment, efficient allocation of wireless channels is consequential to the system performance. This paper presents a distributed channel allocation algorithm based on the concept of dynamic channel boundaries. Under this scheme, each mobile service station (MSS) is assigned a contiguous range of channels not overlapping with those of its neighbors. Channel boundaries between adjacent MSSs are dynamically adjustable through voluntary withdrawals. High efficiency is provided by two important features: early boundary expansion and need-based boundary redrawing. This algorithm imposes a low message volume on the fixed network, has short response times to connection requests and achieves high channel utilization under both evenly and unevenly distributed loads. Channel utilization is further improved if a MSS may borrow channels from its neighbors. Simulation results indicate this algorithm outperforms other distributed channel allocation algorithms.*

## 1. Introduction

In a mobile computing environment, the geographical area is usually divided into hexagonal cells, each serviced by a mobile service station (MSS) at the center of the cell. Figure 1 shows an example of a $7 \times 7$ grid cellular system. When a mobile computing device, called a mobile host (MH), wants to start a communication session, it sends a connection request through a control channel to the nearest MSS. Upon receipt of the request, the MSS searches for a wireless channel which must not be in use in that cell or in neighboring cells, otherwise interference of signals, called co-channel interference, would occur. If such a channel is available, the MSS will inform the MH to use it. Then, the MH will send and receive data packets through the given channel, and the MSS will forward those packets to and from other parts of the wired network.

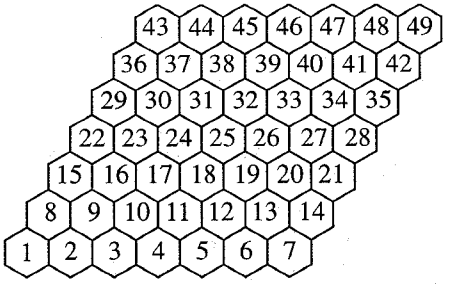


**Figure 1. A 7 x 7 grid cellular system.**

Efficient allocation of wireless channels is crucial in mobile computing. Many channel allocation schemes in the past [5] are centralized in that a central network switch handles all connection requests. Although such schemes have full knowledge of channel usage and can coordinate channel assignments easily, they often create bottlenecks at the links near the central switch, which is also a single point of failure. And they are not very scalable; adding more MSSs or channels may degrade their performance.

Some distributed channel allocation schemes [2, 3] rely on constant monitoring of the signal-to-interference ratio. These schemes are complicated and often require the MHs to participate. When cell sectoring is employed, intra-cell handoffs may result, thus imposing higher burdens on the MHs. According to the two-tier principle [1], participation of MHs in distributed algorithms on mobile network should be minimized because MHs, supported by batteries, have limited computation and communication power. Hence, such channel allocation schemes are undesirable.

The problem to solve is to devise a distributed dynamic channel allocation algorithm running on MSSs that assigns channels to connection requests in a way free of co-channel interferences and denies as few requests as possible so as to maximize the channel utilization. The following requirements must also be satisfied:

1. *Minimal participation of MHs* - Since MHs have limited power, MSSs should select suitable channels on behalf of the MHs. Thus, a MH needs only to



0-8186-7453-9/96 $05.00 

send a connection request to its local MSS and then waits for a reply, which is simply a channel number if one can be found, or a denial message otherwise.

2. *No intra-cell handoffs* - MHs should not be asked to switch to other channels before they leave the current cell. Intra-cell handoffs not only involve more communication between MHs and MSSs, but may also cause short interrupts in communication.

The desirable characteristics of the solution are:

1. Short response time to connection requests,
2. Low message volume imposed on the wired network,
3. Adaptability to fluctuating even and uneven loads.

For the sake of simplicity, it is assumed that:

1. A direct wired connection exists between every pair of adjacent MSSs.
2. A 3-cell cluster system is in effect. That is, a channel cannot be used to support multiple sessions in the same cell and in its immediately neighboring cells, but it can be reused in cells at least two hops away.
3. Each inter-cell handoff is treated as the end of the session in the original cell and the initiation of a new session in the cell the MH enters. However, it would be desirable if handoffs have higher priority than new connection requests.

In this paper, a distributed channel allocation algorithm based on the concept of dynamic channel boundaries is presented. It demands minimal participation on MHs and imposes a low message volume on the fixed network. It responses very fast to connection requests and achieves high channel utilization under both even and uneven load distributions. Channel utilization can be further improved if a MSS is allowed to borrow channels from its neighbors.

The rest of this paper is organized as follows. Section 2 describes the preliminary concepts of how dynamic channel boundaries can efficiently solve the problem of channel allocation. In Section 3, a distributed channel allocation algorithm employing dynamic channel boundaries is presented. A possible enhancement, channel borrowing, is described in Section 4. Section 5 provides simulation results and compares the performance of this algorithm with other algorithms in terms of message volume, response times, request dropping rates and channel utilization. Finally, conclusions are given in Section 6.

## 2. Preliminaries

In a region of hexagon cells, group numbers 1, 2 and 3 can be assigned to the cells such that adjacent cells get different group numbers (Figure 2(a)). Consider the center cell in Figure 2(b). It belongs to group 1 and is surrounded by three cells in group 2 and three in group 3. To avoid co-channel interference, the center cell must not use any channels being used by any of its six neighbors. Likewise, a group-2 cell cannot use any channels in use in its two

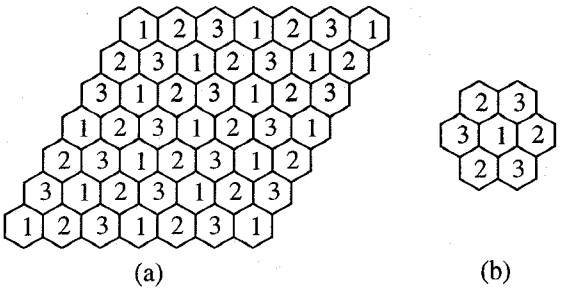


**Figure 2. Assignment of group numbers to cells.**

group-3 neighbors. Thus, the number of channels the center cell may use depends not only on the number of channels being used individually by each of its neighbors, but also on how many channels they occupy collectively. The more they occupy collectively, the less the center cell may use. To reduce the number of channels they occupy and to give the center cell more available channels, MSSs in the same group should use only the channels within the same minimal set, and the channel sets for different groups must be disjoint. This idea can also be applied to other cells to maximize the overall channel utilization.

Concentrating the channels in a channel set is preferred to scattering them over the whole spectrum. Channel management is easier if all channels a MSS may use are within a range. Also, MSSs can communicate their channel usages by giving the bounds of their channel ranges instead of specifying the status of every channel. This helps reduce the message volume on the wired network.

Making use of the two observations above, this paper offers a distributed dynamic channel allocation algorithm based on a mutual exclusion mechanism called dynamic channel boundaries. A dynamic channel boundary is a channel number that separate the ranges of channels two adjacent MSSs may use freely without consulting each other. One MSS uses the channels below the boundary; the other uses the channels at or above it. Channel numbers are considered to wrap around so that the channel immediately above the highest-numbered channel is channel one.

Initially, each MSS is assigned a range of consecutive channels of size about one-third of the spectrum. MSSs in same group share the same channel range, but the channel ranges for different groups are disjoint. The initial boundaries are the lower bounds of the channel ranges. An example of initial assignment for a 100-channel spectrum could be: channels 1-33 assigned to group 1, 34-66 to group 2, and 67-100 to group 3, and the boundaries are 1, 34 and 67. In this way, as long as the number of active sessions in each cell does not exceed the number of channels assigned to it, all connection requests can be satisfied immediately and without any inter-cell communication.

When a station, say $MSS_i$, encounters a heavy load, it sends EXPAND messages to some or all of its neighbors to request a boundary expansion. Upon receiving the

EXPAND messages, the neighbors individually decide how much to withdraw their boundaries with $MSS_i$. After $MSS_i$ collects all the REPLY messages containing their new boundaries, it takes the tightest boundary pair as its new channel range. Since a MSS usually has multiple neighbors, it may have up to three different upper and three different lower boundaries. Clearly, it cannot use a channel outside any of the boundaries without risking channel conflict. If every MSS uses only the channels within its tightest boundary pair and boundaries are moved only by voluntary withdrawals, mutual exclusion to channels and freedom from channel conflict can be guaranteed.

An important advantage of this algorithm is that several expansion requests may proceed concurrently. No requests need to wait and no replies have to be delayed. If two adjacent MSSs try boundary expansions simultaneously, only one of them needs to make a decision and inform the other.

To increase the chance of successful boundary expansions, each MSS should satisfy connection requests with channels closest to its focus point, which is normally the mid-point of its channel range.

MSSs need not wait till all channels are busy to request boundary expansion. If the number of available channels falls below a minimum after filling a connection request, a MSS begins attempting boundary expansion. Before the MSS receives all the replies, it can continue to satisfy further connection requests. Early boundary expansion also reduces the response times to most connection requests.

When a connection request arrives and all channels are busy, the MSS initiates an urgent boundary expansion by sending REQUEST messages to all its neighbors, who will try to withdraw their boundaries by at least one channel.

To help make better decisions on boundary movements, each MSS keep a growth rate, defined as the number of connection requests minus the number of sessions finished in a certain time interval. By conveying the growth rates and the number of available channels, adjacent MSSs can agree on a new boundary that suits the needs of both sides.

Under heavy load, a MSS with extra available channels can shrink its boundaries voluntarily by sending SHRINK messages containing new boundaries to its neighbors, who may then expand their boundaries without requesting.

Under varying loads in different parts of the network, the channel boundaries of peers, nearby MSSs in the same group, may extend in opposite directions and their channel ranges will no longer overlap. This defeats the purpose of dynamic boundaries and degrades channel utilization. To avoid this problem, the channel ranges of the lightly-loaded peers should stay within that of the leader, which is a heavily-loaded peer. If a MSS believes it is the leader, it broadcasts its load and focus point to its peers through its neighbors whenever it expands or shrinks its boundaries. This information allows the peers to decide who the leader is and how to adjust their boundaries to follow the leader.

## 3. The Algorithm

Let NCH be the number of channels in the spectrum. Modulo-NCH arithmetic is assumed. In this algorithm, each MSS maintains the distributed data structures below:

- $Busy_i$: a set representing the channels being used locally. This set can be implemented as a sequence of bits, with each bit indicating whether a channel is busy or not.
- $LowerBound_i$: the lower bound of the usable channel range.
- $UpperBound_i$: the upper bound of the usable channel range. The channels $MSS_i$ may use freely are from $LowerBound_i$ through $UpperBound_i - 1$, inclusively.
- $Focus_i$: the desired point of concentration of busy channels.
- $NbrBound_{ij}$: the channel boundary with the neighbor $MSS_j$.
- $NbrNavail_{ij}$: the last known number of available channels in the neighbor $MSS_j$.
- $Leader_i$: the current leader among the peers, excluding $MSS_i$.
- $LeaderLoad_i$: the load of the current leader.
- $LeaderFocus_i$: the focus point of the current leader.
- $Expanding_i$: indicates if $MSS_i$ is trying boundary expansion.
- $GrowthRate_i$: the number of connection requests minus the number of disconnections in the past INTERVAL seconds. Details of maintaining this variable are omitted in this paper.

All MSSs share the following predefined parameters (the number in the parenthesis after each parameter is the value used in the simulation to be described in Section 5):

- LOW_THRES (2): the minimum number of channels to be maintained in the channel range to avoid starvation.
- INTERVAL (10): the number of seconds *GrowthRate* is based on.
- EXPAND_RATIO (0.0): when the ratio of the number of available channels in $MSS_i$ to the average number of available channels in its neighbors goes below this parameter, $MSS_i$ initiates a boundary expansion attempt.
- EXPAND_SECS (0): a boundary expansion should start this number of seconds before EXPAND_RATIO is reached.
- SHRINK_RATIO (2.0): when the ratio of the number of available channels in $MSS_i$ to the average number of available channels in its neighbors goes above this parameter, $MSS_i$ will shrink its boundaries voluntarily.
- SHRINK_SECS (1): the number of seconds after which the number of available channels in a MSS shrinking its boundaries and the average number of available channels in its neighbors should be balanced.
- SHRINK_LIMIT (1): a MSS shrinks its boundaries only if the ratio of twice the average number of available channels in the neighbors to NCH is less than this parameter.
- LEADER_LIMIT (0.2): the min. ratio of the number of busy channels to NCH for a MSS to consider itself the leader.
- FOLLOW_LIMIT (0.06): the min. ratio of the number of available channels in a MSS to NCH that the MSS should follow the leader, if it is not the leader.

To simplify the explanation of the algorithm, the following formulae are defined:

- $Group_i = (Row_i + Column_i \times 2) \bmod 3 + 1$, where $Row_i$ and $Column_i$ are the row and column position of $MSS_i$ in the grid.
- $Above_{ij} = ((Group_j - Group_i) \bmod 3 = 1)$ is true if the channel range of $MSS_j$ is above that of $MSS_i$.

- $Below_{ij}$ = (($Group_i - Group_j$) mod 3 = 1) is true if the channel range of $MSS_j$ is below that of $MSS_i$.
- $Navail_i = UpperBound_i - LowerBound_i - |Busy_i|$ is the number of available channels in the channel range of $MSS_i$.
- $AvgNbrNavail_i = (\sum_j NbrNavail_{ij}) / Nnbr_i$ is the average number of average channels among the neighbors of $MSS_i$, where $Nnbr_i$ is the number of neighbors $MSS_i$ has.
- $Load_i = |Busy_i|$ is the load of $MSS_i$.
- $IsLeader_i = (Load_i \geq LeaderLoad_i)$ is true when $MSS_i$ considers itself the leader.
- $FollowLeader_i$ = ($Load_i < LeaderLoad_i$ AND $Leader_i \neq 0$ AND $Navail_i \geq$ NCH × FOLLOW_LIMIT) is true when $MSS_i$ decides it should follow the leader.

The algorithm is described below:

(A) Initialization: At the beginning, each MSS is assigned about one third of the NCH channels.

$Busy_i \leftarrow \emptyset$ ;
$LowerBound_i \leftarrow \lfloor NCH \times (Group_i - 1) / 3 \rfloor + 1$;
$UpperBound_i \leftarrow \lfloor NCH \times Group_i / 3 \rfloor + 1$;
$Focus_i \leftarrow \lfloor (LowerBound_i + UpperBound_i) / 2 \rfloor$ ;
$Leader_i \leftarrow 0$;
$LeaderLoad_i \leftarrow \lfloor NCH \times LEADER\_LIMIT \rfloor$ ;
$Expanding_i \leftarrow 0$;
For each neighbor $MSS_j$ of $MSS_i$:
$NbrNavail_{ij} \leftarrow \lfloor NCH / 3 \rfloor + \lfloor Group_j / 3 \rfloor$ ;
If $Above_{ij}$, then $NbrBound_{ij} \leftarrow UpperBound_i$;
else $NbrBound_{ij} \leftarrow LowerBound_i$.

(B) When a session needs to be set-up in cell $C_i$:

1. If $Navail_i = 0$, then
   If $Expanding_i = 0$, initiate the boundary expansion process defined in (D) with a parameter value of −1;
   Wait till $Navail_i > 0$ OR $Expanding_i = 0$;
2. If $Navail_i > 0$, then
   Select a free channel $c$ from ($LowerBound_i$, $UpperBound_i$ − 1) − $Busy_i$ and closest to $Focus_i$ to set-up the session;
   $Busy_i \leftarrow Busy_i \cup \{c\}$;
   else Drop this connection request.
3. If $Navail_i - GrowthRate_i \times$ EXPAND_SECS < $AvgNbrNavail_i \times$ EXPAND_RATIO AND $Expanding_i = 0$, then
   Initiate the boundary expansion process defined in (D) with $Navail_i$ as the parameter, but continue honoring subsequent connection requests until blocked by step 1 above.

(C) When a session using channel $c$ terminates in cell $C_i$:

1. $Busy_i \leftarrow Busy_i - \{c\}$;
2. If $Expanding_i = 0$ AND $AvgNbrNavail_i \times 2 <$ NCH × SHRINK_LIMIT AND $Navail_i - GrowthRate_i \times$ SHRINK_SECS > $AvgNbrNavail_i \times$ SHRINK_RATIO, then
   To achieve about equal number of available channels with the neighbors after SHRINK_SECS, the number of channels to withdraw the boundary is:
   $CutAmount_i \leftarrow (Navail_i - AvgNbrNavail_i - GrowthRate_i \times$ SHRINK_SECS) / 2;
   else $CutAmount_i \leftarrow 0$;
3. If $CutAmount_i \leq 0$, then skip steps 4 to 11.
4. Out of the two groups of neighbors, select the one group of neighbors whose average number of available channels as stored in $NbrNavail_{ij}$ is less than the other group. Let one of the selected neighbors be $MSS_k$.
5. Tighten the boundaries with neighbors selected in step 4 to either $LowerBound_i$ or $UpperBound_i$ as follows:
   If $Above_{ik}$, then for each selected neighbor $MSS_j$,
   $NbrBound_{ij} \leftarrow UpperBound_i$;
   else for each selected neighbor $MSS_j$,
   $NbrBound_{ij} \leftarrow LowerBound_i$;
6. Recalculate the extent of the withdrawal after initial shrinking:
   $CutAmount_i \leftarrow (Navail_i - AvgNbrNavail_i - GrowthRate_i \times$ SHRINK_SECS) / 2;
7. If $CutAmount_i \leq 0$, then go to step 11.
8. If $Above_{ik}$, then
   While ($CutAmount_i > 0$ AND $UpperBound_i - LowerBound_i >$ LOW_THRES AND $UpperBound_i \notin Busy_i$):
   $UpperBound_i \leftarrow UpperBound_i - 1$;
   $CutAmount_i \leftarrow CutAmount_i - 1$;
   else
   While ($CutAmount_i > 0$ AND $UpperBound_i - LowerBound_i >$ LOW_THRES AND $LowerBound_i - 1 \notin Busy_i$):
   $LowerBound_i \leftarrow LowerBound_i + 1$;
   $CutAmount_i \leftarrow CutAmount_i - 1$;
9. Perform the Refocus procedure as defined in (H) on $MSS_i$ with a parameter value of 0.
10. For each selected neighbor $MSS_j$ (including $MSS_k$),
    If $Above_{ik}$, then $NbrBound_{ij} \leftarrow UpperBound_i$;
    else $NbrBound_{ij} \leftarrow LowerBound_i$;
11. $Broadcast_i \leftarrow 2$; Send SHRINK ($NbrBound_{ij}$, $Navail_i$, $Broadcast_i$, $Focus_i$, $Load_i$) messages to $MSS_j$.

(D) Boundary expansion process - let the calling parameter value be $Nremain_i$. When called by steps (B).1 or (B).3, the following actions are taken:

1. $Expanding_i \leftarrow 1$;
2. If $IsLeader_i$ is true, then
   If $Nremain_i < 0$, then $Broadcast_i \leftarrow 1$;
   else $Broadcast_i \leftarrow 2$;
   else $Broadcast_i \leftarrow 0$;
3. If $Nremain_i < 0$, then
   Send EXPAND ($Nremain_i$, $GrowthRate_i$, $Broadcast_i$, $Focus_i$, $Load_i$) messages to all neighbors;
   else
   Out of the two groups of neighbors, select the one group of neighbors whose average number of available channels as stored in $NbrNavail_{ij}$ is more than the other group; Send EXPAND ($Nremain_i$, $GrowthRate_i$, $Broadcast_i$, $Focus_i$, $Load_i$) messages to the selected group of neighbors;
4. $MSS_j$ waits for REPLY messages from its neighbors. For each REPLY ($NewBound_{ji}$, $Navail_j$) message it receives:
   $NbrBound_{ij} \leftarrow NewBound_{ji}$;
   $NbrNavail_{ij} \leftarrow Navail_j$;
5. When $MSS_i$ receives all REPLY messages from its neighbors:
   Among neighbors $MSS_j$ where $Below_{ij}$ is true, take the $NbrBound_{ij}$ closest to $Focus_i$ in the downward direction as the new value for $LowerBound_i$;
   Among neighbors $MSS_j$ where $Above_{ij}$ is true, take the $NbrBound_{ij}$ closest to $Focus_i$ in the upward direction as the new value for $UpperBound_i$;
6. Perform the Refocus procedure as defined in (H) on $MSS_i$ with a parameter value of 0.
7. $Expanding_i \leftarrow 0$;
   Terminate this boundary expansion process.

(E) When $MSS_j$ receives an EXPAND ($Nremain_i$, $GrowthRate_i$, $Broadcast_i$, $Focus_i$, $Load_i$) message from $MSS_i$, $MSS_j$ takes the following actions:

1. $NbrNavail_{ji} \leftarrow Nremain_i$;
2. If $Expanding_j = 1$ AND $MSS_i$ is among the neighbors $MSS_j$ sends EXPAND messages to AND $MSS_j$ has not received the REPLY message from $MSS_i$, then
   If $j < i$, then go to step 11;
   else count the REPLY message from $MSS_i$ as already received.
3. To divide the available channels in $MSS_i$ and $MSS_j$ between them at the same ratio as $GrowthRate_i$ to $GrowthRate_j$, the number of channels $MSS_j$ needs to withdraw its boundary with $MSS_i$ is:
   If $GrowthRate_i + GrowthRate_j \neq 0$, then
   $CutAmount_j \leftarrow (GrowthRate_i \times Navail_j - GrowthRate_j \times Nremain_i) / (GrowthRate_i + GrowthRate_j)$;
   else $CutAmount_j \leftarrow 0$;
   If $CutAmount_j < 0$, then $CutAmount_j \leftarrow 0$;
   If $CutAmount_j > Navail_j$, then $CutAmount_j \leftarrow Navail_j$;
4. If $Nremain_i + CutAmount_j < 0$, then
   $CutAmount_j \leftarrow -Nremain_i$; /* to meet the urgent need of $MSS_i$*/
5. If $CutAmount_j = 0$, go to step 11.
6. While ($CutAmount_j > 0$ AND (if $Below_{ji}$ then $NbrBound_{ji}$ else $NbrBound_{ji} - 1$) $\notin Busy_j$ AND $UpperBound_j - LowerBound_j$ > LOW_THRES):
   Move $NbrBound_{ji}$ toward $Focus_j$ by one channel;
   $CutAmount_j \leftarrow CutAmount_j - 1$;
7. If $Below_{ji}$ is true, then
   If $NbrBound_{ji}$ is closer than $LowerBound_j$ to $UpperBound$, then $LowerBound_j \leftarrow NbrBound_{ji}$;
   else
   If $NbrBound_{ji}$ is closer than $UpperBound_j$ to $LowerBound_j$, then $UpperBound_j \leftarrow NbrBound_{ji}$;
8. If $Above_{ji}$ is true, then $FocusShift_j \leftarrow -\lfloor CutAmount_j / 2 \rfloor$
   else $FocusShift_j \leftarrow \lfloor CutAmount_j / 2 \rfloor$
9. Perform the Refocus procedure defined in (H) on $MSS_j$ with $FocusShift_j$ as the parameter.
10. $MSS_j$ sends a REPLY ($NbrBound_{ji}$, $Navail_j$) to $MSS_i$.
11. Perform the Inform procedure defined in (I).

(F) When $MSS_k$ receives an INFORM ($i$, $Focus_i$, $Load_i$) message:

If $i = Leader_k$ OR $Load_i > LeaderLoad_k$, then
$Leader_k \leftarrow i$;
$LeaderLoad_k \leftarrow Load_i$;
$LeaderFocus_k \leftarrow Focus_i$;
Perform the Refocus procedure defined in (H) on $MSS_k$ with a parameter value of 0.

(G) When $MSS_j$ receives from $MSS_i$ a SHRINK ( $NewBound_{ij}$, $Navail_i$, $Broadcast_i$, $Focus_i$, $Load_i$) message:

1. $NbrBound_{ji} \leftarrow NewBound_{ij}$;
   $NbrNavail_{ji} \leftarrow Navail_i$;
2. If $Below_{ji}$ is true, then
   Among neighbors $MSS_k$ where $Below_{jk}$ is true, take the $NbrBound_{jk}$ closest to $Focus_j$ in the downward direction as the new value for $LowerBound_j$;
   else
   Among neighbors $MSS_k$ where $Above_{jk}$ is true, take the $NbrBound_{jk}$ closest to $Focus_j$ in the upward direction as the new value for $UpperBound_j$.
3. Perform the Refocus procedure as defined in (H) on $MSS_j$ with a parameter value of 0.
4. Perform the Inform procedure as defined in (I).

(H) Refocus procedure - let the procedure be performed on $MSS_m$ and let $FocusShift_m$ be the calling parameter value:

1. If $FollowLeader_m$ is false, then
   $Focus_m \leftarrow \lfloor (LowerBound_m + UpperBound_m + FocusShift_m)/2 \rfloor$
2. If $Focus_m$ is outside the range of ($LowerBound_m$, $UpperBound_m - 1$), then set $Focus_m$ to either $LowerBound_m$ or $UpperBound_m - 1$, whichever is closer to $Focus_m$.

(I) Inform procedure - $MSS_j$ takes the following actions:

1. If $Broadcast_i > 0$, then
   Send an INFORM ($i$, $Focus_i$, $Load_i$) message to the peer of $MSS_i$ which is also the neighbor of $MSS_j$ and is clockwise next to $MSS_i$, if such a peer exists.
2. If $Broadcast_i = 2$, then
   Send an INFORM ($i$, $Focus_i$, $Load_i$) message to the peer of $MSS_i$ which is also the neighbor of $MSS_j$ and is anti-clockwise next to $MSS_i$, if such a peer exists.

## 4. Channel Borrowing

In the algorithm described in Section 3, a connection request will not be denied before a boundary expansion is attempted. But this does not mean denial is necessary. The connection request could have been satisfied by some other channels not being used by the neighbors if the usable channels in a cell were not limited to the channel range of the local MSS.

Consider a connection request arriving at $MSS_i$ that runs out of available channels in its channel range. $MSS_i$ sends to all its neighbors EXPAND messages with $Nremain_i = -1$, telling the neighbors to withdraw their boundaries with $MSS_i$ by at least one channel if possible. Among $MSS_i$'s neighbors, two or more of them share with $MSS_i$ the boundaries that are tightest for $MSS_i$. If their channels at those boundaries are busy, they cannot withdraw their boundaries at all. Thus, the connection request is doomed to be denied, no matter how many available channels exist in the neighborhood of $MSS_i$. The consequence is less-than-optimal channel utilization.

The problem above can be solved by allowing $MSS_i$ to borrow channels from its neighbors if it has a connection request waiting. To allow $MSS_i$ be aware of which channels are available for borrowing, its neighbors can include in each REPLY messages a list of such channels and current lower and upper bounds. It is preferable to limit the length of an available channel list so that the size of a REPLY message is not too long. From the extra information included in the REPLY messages, $MSS_i$ can

determine which channel, if any, it may borrow. $MSS_i$ then sends BORROW messages to inform its neighbors the channel it chooses to borrow. When the communication session which uses the borrowed channel terminates, $MSS_i$ returns that channel to its neighbors by sending them RETURN messages.

Since the borrowed channel is not within the channel range of $MSS_i$, the disjoint property of channel ranges can no longer protect that channel from being used by the neighbors of $MSS_i$. This calls for a new protocol that guarantees mutual exclusion to borrowed channels. At the same time, the protocol should not be too restrictive as to prevent two neighbors of $MSS_i$ not adjacent to each other from borrowing the same channel from $MSS_i$.

The approach for mutual exclusion of borrowed channels is as follows. At the lender side, channels are put on hold once they are quoted in a REPLY message and are released when the BORROW message arrives. The borrower must send BORROW messages to all its neighbors, even if it no longer needs a channel or cannot find a channel to borrow. If a BORROW message contains a channel number, all neighbors of the borrower, not just the lenders, will mark that channel as lent. Channels put on hold or marked as lent are never used locally, but may still be held by and lent to neighbors. To prevent two adjacent MSSs from borrowing the same channel at the same time, a potential borrower will give up borrowing if it finds any of its neighbor attempting boundary expansion concurrently. Details of this protocol is given in the descriptions below.

To support channel borrowing, each $MSS_i$ maintains some additional data structures:

- $HoldCount_{ic}$: the number of holds put on channel $c$.
- $LendCount_{ic}$: the number of neighbors channel $c$ is lent to.
- $HoldList_{ij}$: the set of channels $MSS_i$ is holding for $MSS_j$.
- $Borrow_i$: the set of channels $MSS_i$ has borrowed.
- $Borrowing_i$: indicates if $MSS_i$ is trying to borrow a channel.
- $BorrowSafe_i$: indicates whether $MSS_i$ may safely borrow a channel. Borrowing is unsafe when any neighbor of $MSS_i$ is expanding its boundary at the same time.

A new algorithmic parameter is introduced:

- HOLD_LIMIT (0.06): the ratio of the max. number of channels quoted in a REPLY message to NCH.

Two formulae need to be modified accordingly:

- $Navail_i = UpperBound_i - LowerBound_i - |Busy_i| - |HoldLent_i|$ is the number of available channels in the channel range of $MSS_i$, where $HoldLent_i = \{c \in (LowerBound_i, UpperBound_i - 1) \mid HoldCount_{ic} > 0 \text{ OR } LendCount_{ic} > 0\}$
- $Load_i = |Busy_i| + |Borrow_i|$ is the load of $MSS_i$.

Channel borrowing is implemented by adding the following actions to the algorithm described in Section 3:

1. In the initialization in step (A),
   $Borrowing_i \leftarrow$ false;
   $Borrow_i \leftarrow \emptyset$ ;
   For each channel $c$ in the spectrum:
   $HoldCount_{ic} \leftarrow 0$;
   $LendCount_{ic} \leftarrow 0$;
   For each neighbor $MSS_j$ of $MSS_i$:
   $HoldList_{ij} \leftarrow \emptyset$ ;
2. In step (B).1, before initiating the boundary expansion process,
   $Borrowing_i \leftarrow$ true;
   $BorrowSafe_i \leftarrow$ true;
3. In step (E).1,
   If $Borrowing_j$, then $BorrowSafe_j \leftarrow$ false.
4. In step (E).4,
   If $Nremain_i + CutAmount_j \geq 0$, then $HoldList_{ji} \leftarrow \emptyset$ ;
   else
   $HoldList_{ji} \leftarrow (LowerBound_j, UpperBound_j - 1) - Busy_j$;
   If $HoldList_{ji}$ contains more than HOLD_LIMIT × NCH channels, then
   the extra channels farthest away from the boundary between $MSS_i$ and $MSS_j$ are removed from $HoldList_{ji}$.
   For each channel $c$ in $HoldList_{ji}$:
   $HoldCount_{jc} \leftarrow HoldCount_{jc} + 1$;
5. In step (E).10, $HoldList_{ji}$, $LowerBound_j$ and $UpperBound_j$ are included in the REPLY message sent to $MSS_i$.
6. Step (B).2 is rewritten as:
   If $Navail_i > 0$, then
   Select a free channel $c$ from $(LowerBound_i, UpperBound_i - 1) - Busy_i$, for which $HoldCount_{ic}$ and $LendCount_{ic}$ are both zero, and closest to $Focus_i$ to set-up the session;
   $Busy_i \leftarrow Busy_i \cup \{c\}$;
   else
   If $Borrowing_i$ AND $BorrowSafe_i$ AND there exists a channel $c$ in any of the $HoldList_{ji}$ contained in the REPLY messages such that for each neighbor $MSS_j$ of $MSS_i$, channel $c$ is either included in $HoldList_{ji}$ or is outside $(LowerBound_j, UpperBound_j - 1)$, then
   Pick such a channel $c$ closest to $LowerBound_i$ or $UpperBound_i - 1$ to set-up the session;
   $Borrow_i \leftarrow Borrow_i \cup \{c\}$
   else
   $c \leftarrow 0$;
   Drop this connection request.
7. A new step (B).2a is inserted between steps (B).2 and (B).3:
   If $Borrowing_i$, then
   If $c \notin Borrow_i$, then $c \leftarrow 0$;
   $MSS_i$ sends BORROW ($c$) messages to all neighbors;
   $Borrowing_i \leftarrow$ false.
8. When $MSS_j$ receives a BORROW ($c$) message from $MSS_i$:
   For each channel $c'$ in $HoldList_{ji}$:
   $HoldCount_{jc'} \leftarrow HoldCount_{jc'} - 1$;
   $HoldList_{ji} \leftarrow \emptyset$;
   If $c \neq 0$, then $LendCount_{jc} \leftarrow LendCount_{jc} + 1$.
9. When a communication session terminates in step (C).1,
   If $c \in Borrow_i$, then
   Send RETURN ($c$) messages to all neighbors of $MSS_i$;
   $Borrow_i \leftarrow Borrow_i - \{c\}$;
10. When $MSS_j$ receives a RETURN ($c$) message from $MSS_i$:
    $LendCount_{jc} \leftarrow LendCount_{jc} - 1$.

## 5. Simulation Environment and Results

A 7×7 grid of hexagonal cells with a 3-cell cluster system and a 100-channel spectrum (NCH=100) is used. The duration of a communication session is exponentially distributed with a mean of 3 minutes. The channel request arrival rate in each cell follows a Poisson distribution. Its mean is varied to generate varying channel request loads at different times and cells. Algorithmic parameters are set to the values as given in Sections 3 and 4.

Assumed that composing and sending a message takes 2 milliseconds (msec) at the sender (MH or MSS), and receiving and processing it takes another 2 msec at the receiver. Message propagation times are absorbed in these 4 msec. Here, response time is defined as the elapsed time between a MH sending a connection request and its receiving a channel number or a denial message. Hence, the minimum response time under this model is 8 msec.

To account for other workload of the MSSs unrelated to channel allocation, say routing data packets, it is assumed a MSS spends on such workload 0.5% of its time, evenly distributed, for every active session. For example, a MSS with 50 active connections spends one out of every four msec on such workload. This timing model is more realistic than if the MSSs are dedicated to channel allocation.

Simulations are carried out to compare the performance of the algorithm in this paper (with/without channel borrowing) with two other algorithms: a distributed DCA algorithm proposed by Prakash, Shivaratri, and Singhal [4] and a fixed channel allocation algorithm. In the fixed algorithm, each MSS can use only the pre-assigned, conflict-free channels (of size one-third of the spectrum). Hence, no inter-cell communication is required. It is included to see how different dynamic algorithms improve over the fixed one under various load distributions.

The simulation results are presented in Figures 3. The four charts on the left show how the algorithms perform under evenly distributed loads, and the right ones are for uneven load distributions. The data in each chart are collected from ten runs for each of the four algorithms and each run consists of 100,000 channel requests.

Figure 3 shows that the basic algorithm (without channel borrowing) presented in this paper has nearly optimal response times even under heavy channel request load. This is due to pre-allocation of channels and early boundary expansion. It imposes an extremely low message volume on the fixed network. When the offered load is close to 100%, less than two messages are sent per connection request. Only about 4% of all connection requests are dropped under a full load, thus achieving a channel utilization of 93.5% (91.5% for uneven load) of the total capacity. The high channel utilization under both evenly and unevenly distributed load is the benefit of employing the concept of dynamic channel boundaries. It is worth noting that the simple fixed channel allocation algorithm also attains a comparable channel utilization under evenly distributed load, but without imposing any burden on the wired network or incurring any response delay.

With channel borrowing added to the algorithm, the peak request denial rates is reduced to 3% for evenly distributed load and 2.5% for unevenly distributed load. The corresponding channel utilizations rise to 94% and 93% respectively. These figures represent the best performance among the four algorithms under simulation. The costs of this high performance are an increase of one msec in response time and on average one additional message on the fixed network per connection request under a full load.

## 6. Conclusion

This paper presents a distributed channel allocation algorithm based on the concept of dynamic channel boundaries. Simulation results show that the algorithm achieves very high channel utilization while incurring a very low message volume and a slight penalty on response times. Channel utilization is further improved when channel borrowing is incorporated. This algorithm outperforms two others in the simulation experiments.

To overcome adjacent channel interference, the algorithm in this paper can easily be modified by mapping adjacent frequencies to channel numbers far apart. For example, the frequencies $f_1, f_2, f_3, f_4, \ldots, f_{99}, f_{100}$ in a 100-channel system can be mapped in a interleaved manner to channels 1, 51, 2, 52, . . . , 50, 100. In this way, as long as a MSS do not occupy more than half of all the channels, adjacent interference will not occur.

## References


[1] B. R. Badrinath, A. Acharya, and T. Imielinski. Designing Distributed Algorithms for Mobile Computing Networks. In *Proceedings of the 14th International Conference on Distributed Computing Systems*. May 1994.

[2] N. Bambos and G. J. Pottie. On the Maximum Capacity of Power-controlled Cellular Networks. In *Wireless Communications: Future Directions*. Edited by J. M. Holtzman and D. G. Goodman, Boston: Kluwer, 1993.

[3] J. C.-I. Chuang and N. R. Sollenberger. Performance of Autonomous Dynamic Channel Assignment and Power Control for TDMA/FDMA Wireless Access. *IEEE Journal on Selected Areas in Communications*, 12(8):1314-1323, October 1994.

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

[5] J. F. Vucetic and D. D. Dimitrijevic. Adaptive Channel Allocation in Cellular Networks with Multi-user Platforms. In *Wireless Communications: Future Directions*. Edited by J. M. Holtzman and D. G. Goodman, Boston: Kluwer, 1993.

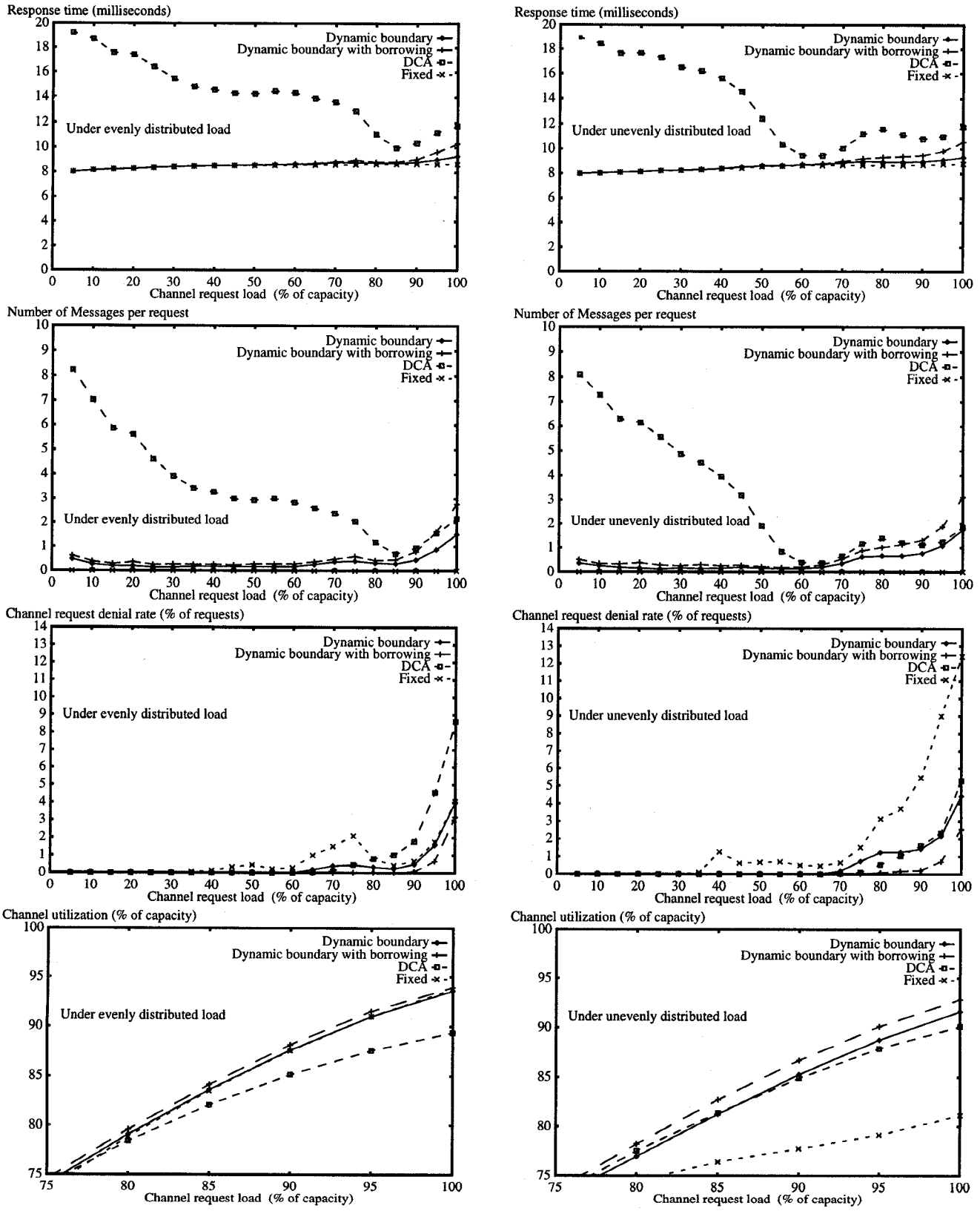


**Figure 3. Simulation results.**