Adaptive Distributed Dynamic Channel Allocation for Wireless Networks*

Anurag Kahol, Sumit Khurana, Sandeep K Gupta and Pradip K Srimani
Department of Computer Science
Colorado State University
Ft. Collins, Colorado, U.S.A.
{kahol, khurana, gupta, srimani}@cs.ColoState.edu

May 27, 1998

Technical Report CS-98-105

*To appear in the **Proceedings of ICPP Workshop on Wireless Networks and Mobile Computing**, Minneapolis, August 10-14, 1998

Adaptive Distributed Dynamic Channel Allocation for Wireless Networks[*]

Anurag Kahol, Sumit Khurana, Sandeep K Gupta and Pradip K Srimani
Department of Computer Science
Colorado State University
Ft. Collins, Colorado, U.S.A.
{kahol, khurana, gupta, srimani}@cs.ColoState.edu

Abstract

Channel allocation schemes in a mobile computing (wireless) environment can be either static or dynamic. Static allocation offers negligible channel acquisition time and zero message complexity and works well at a low system load; the performance steadily decreases as system load increases since many calls are dropped; in case of even temporary hot spots many calls may be dropped by a heavily loaded switching station even when there are enough idle channels in the interference region of that station. On the other hand, dynamic schemes provide better utilization of the channels at higher loads albeit at the cost of higher channel acquisition time and some additional control messages. Our purpose in the present paper is to propose a combined channel allocation scheme that each switching station can tune to its own load independent of other stations in its interference region; the objective is to minimize the call drop rate and at the same time maintain a minimum average channel acquisition time and minimum control message complexity.

1. Introduction

The radio spectrum is a scarce resource in wireless networks. Further, radio signals are susceptible to interference and multipath fading. Thus, efficient use of radio bandwidth is essential for supporting large number of mobile users in a mobile computing environment. In order to reuse radio spectrum, the current wireless systems use a cellular architecture; the geographical area is divided into several coverage areas called *cells*, and the radio spectrum is divided into a number of wireless communication *channels* using combination of frequency division, time division, and code division techniques [5]. A channel r can be used by a cell c without any *interference* if it is not concurrently used by any other cell within the *minimum reuse distance* of cell c. The problem of channel allocation is to devise a scheme for allocating channels to cells in the system so as to eliminate channel interference; the design objectives are to minimize *channel acquisition time*, to minimize the number of calls that are denied service, and to minimize the *control message complexity*.

Channel allocation schemes are of two types: *fixed* (or static) [7, 1] and *dynamic* [10, 9, 2]. Fixed channel allocation techniques assign channels to cells according to some a-priori known reuse patterns and these assignments don't change with time or load in the system. These techniques are simple to implement and have negligible channel acquisition time. However, they do not adapt to changes in traffic patterns and mobile user distribution which results in poor utilization of wireless bandwidth as well as denial of service even when there are idle channels in the interference region of the cell. In a dynamic channel allocation scheme, all the channels are placed in a pool and they are assigned to the cells as and when they are needed such that there is no channel interference. Dynamic allocation schemes improves bandwidth utilization and minimizes call drops at the cost of increased channel acquisition time and additional message complexity (arbitration between cells in a neighborhood needs exchange of control messages and hence extra time).

Fixed channel allocation schemes work well at uniform loads [6]. This is because the assignments of channels to cells is made in such a way that a channel can be reused in other cells at distances greater than the interference region of a particular cell. This allocation does not change over time. As compared to this, dynamic schemes can allocate channels to cells as and when they are needed. Dynamic schemes also work well under moderate system loads when a few cells can become highly loaded for short periods of time. This is because if a particular cell becomes a hot-spot for a short duration it can use any channel not in use in it's interference region and can thus support calls which it would otherwise have had to drop. The purpose of a hybrid scheme, that attempts to combine the advantages of the static and the dynamic schemes is to minimize the call drop rate and at the same time minimize the channel acquisition time. There exist a number of such hybrid schemes in the literature [4].

[*]To appear in the **Proceedings of ICPP Workshop on Wireless Networks and Mobile Computing**, Minneapolis, August 10-14, 1998

In this paper, we propose a new hybrid scheme for channel allocation in a mobile environment; the algorithm is distributed and adaptive in the sense that each node (cell) in the system continually monitors its own load (rate of request for channels) and switches back and forth between static and dynamic modes of allocation depending on threshold values of loads (these threshold values are used to fine tune the overall performance of the system) and each node adapts to its own load independent of the other nodes in its interference region. Thus, when the load is relatively low at a node, it enjoys negligible small channel acquisition time and no message complexity; when the load at a node is high for a short period of time, it can still borrow unused channels from its neighbors in the interference region, and thus reduce the call drop at the cost of some control messages and the channel acquisition time is increased to some extent. We provide some empirical performance study of the algorithm and compare it with some existing schemes.

2. Preliminaries

2.1. System Model

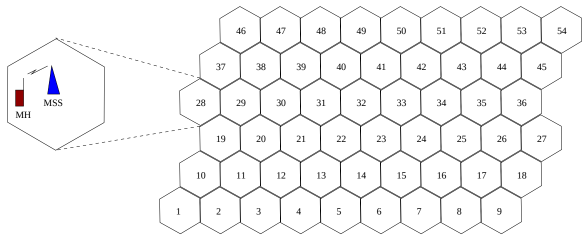

Figure 1. Cellular Communication Architecture

In this paper, we consider the problem of channel allocation for a cellular communication network. A cellular communication network consists of an array of hexagonal cells as shown in Fig.1. We assume that there are $\mathcal{N}$ cells in the system numbered from 1 to $\mathcal{N}$. Each cell except the boundary cells have six neighbors. A mobile service station (MSS) is incharge of a cell. We denote the mobile service station of cell c as MSS$_c$. When a mobile host (MH) in cell c needs to communicate with another MH it first sends a request to MSS$_c$ to acquire a channel using a *control channel*. MSS$_c$ then finds a free channel using a channel allocation procedure and informs the MH which channel it can use. The MH then starts using that channel. We assume that each MH is a mobile computer or a mobile phone. And, a channel can be used for either data or voice communication. After an MH is done using a channel it informs its MSS that it no longer needs that channel. Also when an MH moves out of the cell while it is using a channel, the *handoff* procedure between the MSS of its old cell and the MSS of its new cell ensures that the channels which the MH was using in the old cell are relinquished and new channels are acquired to continue the on going communications in the new cell. The MSS can assign a free channel to another MH in the same cell or relinquish (free) the channel so that it can be used by another cell. The wireless communication bandwidth is divided into n channels. The channels are numbered from 1 to n. We refer to the set of all channel ids as $Spectrum$, i.e. $Spectrum = \{1, 2, \ldots, n\}$. To avoid *co-channel interference*, the distributed channel allocation algorithms discussed in this paper ensure that if a channel r is being used by MSS$_i$ then it is not being simultaneously used for communication in any cell within *minimum reuse distance* from MSS$_i$[1]. The area covered by these cells is called the *interference region* of cell i. We denote the set of Ids of all cells in the interference region of cell i by IN_i.

2.2. Distributed Channel Allocation Schemes

Distributed dynamic channel allocation scheme can be categorized into *update based schemes* and *search based schemes*. In [4], Dong and Lai have presented representative scheme for each approach, namely the *basic update scheme* and the *basic search scheme*. In the basic search scheme a MSS needing a channel searches its interference region for an available channel. This is done by sending a request message to every MSS in the interference region. Each MSS responds by sending its set of used channels. After receiving the response from every MSS in its interference region, an MSS computes the set of available channels and selects one of them. The search procedure ensures that no two MSS in each others interference region simultaneously select the same channel by using timestamps with the request messages. An MSS which is currently searching for a channel defers the response to any request message with a higher timestamp than its request message until it has completed its search.

On the other hand in the basic update scheme, every node maintains information about the channels which are being used in its interference region. When a node needs a channel it selects a free channel r and asks the MSSs in its interference neighborhood for permission to use it. Upon receiving permission to use channel r from all the MSSs in its interference region, the MSS can start using that channel. However, before doing so it sends acquisition message to all the MSS in its interference region so that they can update their local information about the channels being used in their interference region. Further, when an MSS is done using a channel it sends a release message to all the MSS in its interference region. In order to prevent co-channel interference, the basic update scheme also uses timestamps with its request message. However, in this scheme the request messages are not deferred. While waiting for permission to use channel r from all the MSS in the interference region, if it receives a

[1]Some channel allocation algorithms only try to minimize co-channel interference.

request from another MSS for the same channel r then it responds with reject if this request's timestamp is greater than its request timestamp, otherwise it responds with grant and abort its own request for channel r. In case of failing to acquire channel r, the MSS tries to acquire another channel which is free according to its local information.
Hence, the basic update scheme permits more concurrent channel allocation than the basic search. However, the basic search scheme is better suited under heavy load situation when there are only few available channels. It is worth noting that in basic update scheme there can be large number of attempts to acquire a channel under heavy load. We next present our channel allocation scheme in which an MSS uses update scheme or the search scheme based on the number of free available channels and number of attempts it has made to acquire a channel.

3. Proposed Scheme

3.1. Data Structures

Each MSS$_i$ or node i in the system maintains the following local variables and data structures:

- $\boldsymbol{PR_i}$ is the set of primary channels statically assigned to node i; Each node also has the information of PR_j sets from all node j in IN_i.
- $\boldsymbol{Use_i}$ is the set of all channels currently in use by node i;
- $\boldsymbol{U_j}$, $j \in IN_i$, is the set of all channels used by node j (as known to node i);
- $\boldsymbol{NFC_i}$ is a list of tuples (t, s) which indicates that the number of free primary channels at time t changed to s, $0 \leq s \leq |PR_i|$. This list is is maintained to retrieve the number of free primary channels at time t, $0 \leq t \leq W$, units in past from current time, where W is the window size use to predict the future value of number of free channels.
- $\boldsymbol{I_i}$ is the set of all channels in use by nodes in IN_i;
- $\boldsymbol{mode_i}$ is an integer variable that can assume values from the set $\{0, 1, 2, 3\}$; $mode_i = 0$ indicates that the node is in *local* mode; $mode_i = 1$ indicates that the node is in a *borrowing* mode with no pending update or search request; $mode_i = 2$ indicates that the node is in a *borrowing* mode with a pending update request; and $mode_i = 3$ indicates that the node is in a *borrowing* mode with a pending search request.
- $\boldsymbol{UpdateS_i}$ is a set which contains ids of all nodes in IN_i which have requested for updates from node i.
- $\boldsymbol{DeferQ_i}$ is a local request queue containing those requests arriving from its neighbors which node i has decided not to send response immediately.
- $\boldsymbol{waiting_i}$ is a integer variable which indicates the number of search request messages it has responded to and has not received the corresponding acknowledgment.
- $\boldsymbol{pending_i}$ is a boolean which is true when a node has a pending local request and $waiting_i > 0$.
- $\boldsymbol{rounds}$ is an integer variable which indicates the number of attempts to acquire a channel in the borrowing update mode.

3.2. Message Types

Nodes use following types of messages to communicate with their neighbors. Following is a description of each message type and its purpose.

- **REQUEST(**req_type, r, ts_j, j**)** — indicates that the sending node is requesting to acquire a channel; req_type may be 1 or 2 indicating the nature of the request (update or search), ts_j is the timestamp of the node j at the time of generating the request.
- **RESPONSE**(res_type, j, ch) — indicates that the sending node is responding to the request from the receiving node; res_type can be 0, 1, 2, or 3 indicating the nature of response (reject, grant, search, or status); ch is a channel id if res_type is 0 or 1 (i.e. reject or grant); ch is Use_j if res_type is 2 or 3 (i.e. search or status).
- **CHANGE_MODE(**$mode$**,**j**)** — indicates the mode of sending node j has changed; $mode$ can be 0 of 1 indicating the new mode of node j (local or borrowing).
- **RELEASE(**j, r**)** — indicates that the sending node j has released channel r.
- **ACQUISITION(**acq_type, j, r**)** — indicates that the sending node j has acquired channel r; acq_type may be 0 or 1 indicating the mode in which the channel was acquired (non-search or search).

3.3. System Primitives

The following primitives are used in the pseudo code given in the next SubSection. They are used to simplify the presentation of the algorithm and their functions are described below.

- **enqueue**(*node*) adds a request from *node* to the tail of the request queue $DeferQ_i$.
- **dequeue**() removes the request at the head of the queue $DeferQ_i$ and returns the identity of the request's owner as well as the id of the requested channel.
- **time()** returns the current local time.

- **add_nfc(t, s)** records in NFC_i that the number of free channels at time t was s. Further, it also ensures that any information in NFC_i about number of free channels prior to time $t - W$ is deleted from NFC_i.

- **get_nfc(t)** returns the the number of free primary channels at time t.

3.4. Events

Requesting to Acquire a Channel When a node wants to acquire a new channel, it invokes the procedure Request_Channel which is given in Figure 2.
Receiving a REQUEST Message: When a node receives a REQUEST message from any of its neighbors, it invokes the procedure Receive_Request which is described in Figure 4.
Receiving a CHANGE_MODE message: When a node receives a CHANGE_MODE message from any of its neighbors, it invokes the procedure Receive_Change_Mode which is described in Figure 5.
Receiving an ACQUISITION message: When a node receives a ACQUISITION message from any of its neighbors, it invokes the procedure Receive_Acquisition which is described in Figure 7.
Receiving a RELEASE message: When a node receives a REQUEST message from any of its neighbors, it invokes the procedure Receive_Release which is described in Figure 8.
Deallocating a channel in use: When a node no longer needs a channel it is using currently, it wants to deallocate the channel by invoking the procedure Deallocate which is given in Figure 9.

3.5. Algorithm Description

Each cell i in the system is assigned a set of primary channels PR_i according to some reuse pattern. Further, each cell is initially in the *local* ($mode_i = 0$) channel selection mode. In this mode, as long as the number of channels needed by a cell is less that the size of its primary channel set it can satisfy all the channel requests locally. However, a cell may transiently require more channels than in its PR set. For this purpose every cell in the system computes its rate of channel consumption when there is a change in its channel usage. Whenever the number of free primary channels is estimated to fall below a threshold θ_l based on the channel consumption rate cell in the time it would take to get channel usage information from all the cells in it's interference neighborhood, cell i enters a *borrowing* ($mode_i = 1$) channel selection mode and informs all the cells in its interference region IN_i by sending them a CHANGE_MODE message. The function for performing this mode check is given in Fig. 6. Function check_mode() uses a window of size W to predict the number of free primary channels after time $2 * T$, where T is the maximum time to communicate with another node in the interference region and so $2 * T$ is the round trip delay

```
Procedure Request_Channel($ts_i$);
begin
    if($mode_i = 0$) /* $local$ mode */
    then
        if($waiting_i > 0$) then{
            $pending_i =$ TRUE;
            wait UNTIL $waiting_i = 0$;}
        $pending_i = FALSE$;
        if($\exists r \in (PR_i - (Use_i \cup I_i))$) then {
            acquire($r$); return($r$);}
        else
            check_mode();
            wait UNTIL RESPONSE($3, j, U_j$) is received
                from each $j \in IN_i$;
            return(Request_Channel($ts_i$));
    else /* $borrowing$ mode */
        if($\exists r \in (PR_i - (Use_i \cup I_i))$)
        then{ acquire($r$); return($r$);}
        else
            $j = Best()$; $rounds = rounds + 1$;
            if($j \in IN_i \wedge rounds \leq \alpha \wedge$
                $(\exists r \in PR_j \cap (Spectrum - (Use_i \cup I_i)))$)
            then
                $mode_i = 2$;
                send REQUEST($0, r, ts_i, i$) to every $j \in IN_i$;
                wait UNTIL RESPONSE($G_j, j, r$) is received
                    from each node $j \in IN_i$;
                if($\bigwedge_{j \in IN_i}(G_j = 1)$) then {
                    acquire($r$); return($r$);}
                else
                    $mode_i = 1$;
                    send RELEASE($i, r$) to all node $j \in IN_i$
                        for which $G_j = 1$;
                    return(Request_Channel($ts_i$));
            else
                $mode_i = 3$;
                send REQUEST($1, -1, ts_i, i$) to every $j \in IN_i$;
                wait UNTIL RESPONSE($G_j, j, U_j$) is received
                    from each node $j \in IN_i$;
                $Free_i = Spectrum - Use_i - \bigcup_{j \in IN_i} U_j$;
                if($\exists r \in Free_i$) then{
                    acquire($r$); return($r$);}
                else{
                    acquire(-1); return(-1);}
end
```

Figure 2. Node i needs a channel

```
Function acquire(r)
    if($r \in Spectrum$) then $Use_i = Use_i \cup \{r\}$;
    $rounds = 0$;
    case($mode$)
        0,1: send ACQUISITION($0, i, r$)
            to every $j \in UpdateS_i$;
        2: $mode_i = 1$;
        3: send ACQUISITION($1, i, r$) to every
            $j \in IN_i$; $mode_i = 1$;
    end case;
    while $DeferQ_i \neq \emptyset$ do
        $(req\_type, q, TS, j) = dequeue()$;
        if($req\_type = 0$)
        then /* deferred update request */
            if($r \in Use_i$) then
                send RESPONSE($0, i, q$) to node $j$;
            else {
                send RESPONSE($1, i, q$) to node $j$;
                $I_i = I_i \cup \{q\}$; $U_j = U_j \cup \{q\}$};
        else /* deferred search request */
            $waiting_i = waiting_i + 1$;
            send RESPONSE($2, i, Use_i$) to node $j$;
    end while
    if($mode = 0$) check_mode();
end func;
```

Figure 3. Function Acquire.

to send CHANGE_MODE message to all the nodes in the interference region and get their Use set via RESPONSE message. Further, a cell in a borrowing mode returns to the local mode when its channel consumption rate falls to a value such that the number of free primary channels will exceed a threshold θ_h, in the time it would take to communicate with other cells. ($\theta_l < \theta_h$). Using state dependent threshold value for triggering transition between local and borrowing modes prevents the situation in which a cell jumps back and forth between local and borrowing modes.

Upon receiving the CHANGE_MODE message from cell i, every cell in IN_i enters cells i's id in its local set $UpdateS_i$ and sends its used cell information to cell i. A cell i uses $UpdateS_i$ to inform all cells in IN_i in the borrowing mode about changes in its channel status information. This enables a cell in the borrowing mode to determine which channels are not being currently used in its interference region. In the borrowing mode, a cell tries to borrow a channel from other cells in its interference neighborhood whenever all its primary channels are being used. In order to borrow a channel, cell i selects one of the free channels r, based on it's knowledge of it's own use set Use_i and the set of channels used by other cells in it's interference region I_i. It queries the cells in its interference neighborhood for permission to

```
Procedure Receive_Request($req\_type, r, TS, j$)
begin
    if($req\_type = 0$) /* update request */
    then
        case($mode_i$)
            0,1:
                if($r \in Use_i$) then
                    send RESPONSE($0, i, r$) to node $j$;
                else{ send RESPONSE($1, i, r$) to node $j$;
                    $I_i = I_i \cup \{r\}$; $U_j = U_j \cup \{r\}$;
                    check_mode()};
            2:/* pending update request message */
                if($r \in Use_i \vee ts_i < TS$) then
                    send RESPONSE($0, i, r$) to node $j$;
                else { send RESPONSE($1, i, r$) to node $j$;
                    $I_i = I_i \cup \{r\}$; $U_j = U_j \cup \{r\}$;
                    check_mode()};
            3: /* pending search request message */
                if($ts_i < TS$) then enqueue($j$);
                else {send RESPONSE($1, i, r$) to node $j$;
                    $I_i = I_i \cup \{r\}$; $U_j = U_j \cup \{r\}$;
                    check_mode()};
        end case
    else /* search request */
        case($mode_i$)
            0:
                if($pending_i \wedge ts_i < TS$) then enqueue($j$);
                else{$waiting_i = waiting_i + 1$;
                    send RESPONSE($2, i, Use_i$) to node $j$;}
            1:
                $waiting_i = waiting_i + 1$;
                send RESPONSE($2, i, Use_i$) to node $j$;
            2,3:
                if($ts_i < TS$) then enqueue($j$);
                else {$waiting_i = waiting_i + 1$;
                    send RESPONSE($2, i, Use_i$) to node $j$;}
        end case
end
```

Figure 4. Node i receives REQUEST (req_type, r, TS, j) from node j.

```
Procedure Receive_Change_Mode($mode$,$j$)
begin
    if($mode = 0$)
    then $UpdateS_i = UpdateS_i - \{j\}$;
    else
        $UpdateS_i = UpdateS_i \cup \{j\}$;
        send RESPONSE($3, i, Use_i$) to node $j$;
end
```

Figure 5. Node i receives CHANGE_MODE($mode$,j) from node j.

```
Function check_mode();
begin
    $s = |PR_i - (I_i \cup Use_i)|$;
    $t = time()$;
    $add\_nfc(t, s)$;
    $last = get\_nfc(t - W)$;
    $next = s + 2 * T * (s - last)/W$;
    if($(mode_i = 0) \wedge (next \leq \theta_l)$)
    then
        $mode_i = 1$;
        send CHANGE_MODE($1, i$) to each node $j \in IN_i$;
    else if($(mode_i = 1) \wedge (next \geq \theta_h)$)
        then
            $mode_i = 0$;
            send CHANGE_MODE($0, i$) to each node $j \in IN_i$;
end Func;
```

Figure 6. Node i checks to change its mode.

```
Procedure Receive_Acquisition($acq\_type, j, r$)
begin
    if($r \in Spectrum$) then
        $I_i = I_i \cup \{r\}$;
        $U_j = U_j \cup \{r\}$;
        check_mode();
    if($acq\_type = 1$) $waiting_i = waiting_i - 1$;
end
```

Figure 7. Node i receives ACQUISITION(acq_type, j, r) from node j.

```
Procedure Receive_Release($j, r$)
begin
    $I_i = I_i - \{r\}$;
    $U_j = U_j - \{r\}$;
    check_mode();
end
```

Figure 8. Node i receives RELEASE(j, r) from node j.

```
Procedure Deallocate(r)
begin
    $Use_i = Use_i - \{r\}$
    if($mode_i = 0$)
    then
        send RELEASE($i, r$) to each $j \in UpdateS_i$;
    else
        send RELEASE($i, r$) to each $j \in IN_i$;
        check_mode();
end
```

Figure 9. Node i deallocates channel r ($r \in Use_i$)

use r, since r may have been acquired by another cell since the last update was received by i.

Borrowing of channel r succeeds if cell i receives a grant messages from all the cells in IN_i. All cells j, that sent a grant message to cell i, add cell i's id to their list of interfered channels I_j. If cell i receives a reject RESPONSE message from one or more cell in IN_i it does not acquire r. It sends a RELEASE(r) message to all the cells that had sent it a grant message. On receiving a RELEASE(r) message a cell j removes r from I_j. When cell i is done using channel r it relinquishes r by sending a RELEASE(r) message to all the cells in IN_i. In order to maximize the success probability of channel borrowing, cell i always tries to borrow a channel from a cell in its interference neighborhood which has the least number of neighbors in borrowing mode. The function $Best()$ for determining such a neighboring cell is given in Fig. 10.

Further in order to bound the number of attempts a cell makes to acquire a channel, a cell enters the *borrowing search mode* after α rounds, where α is a parameter of the algorithm denoting maximum number of atempts a cell makes in borrowing update mode (i.e. $mode_i = 2$). In the borrowing search mode ($mode_i = 3$) a cell requesting a channel with timestamp ts defers response message to any other

```
Function Best();
begin
    $NotBorrowing = IN_i - UpdateS_i$;
    $Free = Spectrum - (Use_i \cup I_i)$;
    $min\_id = -1$;
    $min\_bn = \infty$;
    for each $j \in NotBorrowing$
        if($Free \cap (Spectrum - U_j) \neq \emptyset$)
        then
            $common\_bn = |UpdateS_i \cap IN_j|$;
            if($min\_bn > common\_bn$)
            then{$min\_id = j; min\_bn = common\_bn;$}
    end for;
    return($min\_id$);
end Func;
```

Figure 10. Function to determine the neighbor from which node i should borrow a channel.

channel request with a timestamp higher than ts until it gets a chance to select a channel. In response to a REQUEST message from a cell in borrowing search mode a cell sends its set of used channels. Upon receiving the used sets from all the cells in its interference neighborhood a cell computes the set of free channels and select one channel from this set. Hence, the sequentialization of concurrent requests in the interference neighborhood of cell in borrowing search mode guarantees that a cell in the borrowing search mode will acquire a free channel if there is one available.

A request message from a cell i in borrowing search mode may be sent when other cells in it's interference region in local or borrowing update modes are trying to acquire a channel. When a cell in borrowing update mode receives a search REQUEST message with a higher timestamp from cell i, it defers cell i's request till it's own request is completed. Similarly cell i defers all search or update requests with higher timestamps. This ensures that each request has a consistent view of the set of used channels in it's interference region when it chooses a channel to acquire. On acquiring a channel through search, a cell sends each cell in it's interference region an ACQUISITION message. All cells j in IN_i update their channel interference set I_j on receiving the ACQUISITION message. A cell in local mode defers it's local request with a higher timestamp till an ACQUISITION message for the search request with a lower timestamp is received (i.e. till $waiting_i = 0$). Thus all requests from cells in an interference region are sequentialized with respect to their timestamps whenever any cell in that interference region goes into search mode. Even if all cells in an interference region are in borrowing mode but have not exhausted their primary channels a cell going into the search mode will get a channel since the decision is based only on the used sets of all cells. This ensures that there is no unsatisfied request when channels are available.

4 Correctness

Theorem 1 *A channel r is not acquired by two cells that are within the minimum reuse distance of each other.*

Proof: The proof is by contradiction. Assume that channel r is acquired in two cells i and j such that $i \in IN_j$. Since $i \in IN_j$, at least one of the two cells, say i, is a secondary cell of r, i.e. $r \in Spectrum - PR_i$. Further, cell i could have acquired r through either using update or search scheme. We assume that cell i requested for a channel with a lower timestamp than cell j. Hence, we consider the following cases:

1. cell i acquires r in borrowing search mode:
   (a) if cell j is in the borrowing search mode its request is deferred by cell i till it completes its channel allocation. Any channel acquired by cell i, say channel r, will be in the Use_i set which cell i sends to cell j after it completes its search. Hence, cell j will not acquire any channel r which has been already acquired by cell i.
   (b) if cell j is in borrowing update mode and is requesting for channel r, cell i will defer cell j's message till has acquired channel r and then send its channel use set to cell j. Hence, cell j will not be able to acquire channel r.
   (c) if cell j is in local mode then:
      i. If cell i has finished it's request before the request from cell j arrives, cell would have waited for the ACQUISITION($1, r, i$) message from cell i. Thus cell j will not acquire channel r.
      ii. If cell i's request arrives after cell j has already acquired channel r then cell j will send its channel use set to cell i and then cell i will not be able to acquire channel r.
      iii. If cell j's channel request message arrives during after it has send out response to the search request message to cell i and before it receives the ACQUISTION message from cell i, cell j will wait till the ACQUISTION message is received and this guarantees that cell j does not acquire channel r if cell i has already acquired it.
2. cell i acquires channel r in borrowing update mode:
   (a) cell i's request has a lower timestamp than that of cell j's: If cell j is in borrowing search mode it's request is deferred by cell i till cell i completes its channel allocation procedure. If cell i acquires

channel r it will be in the use set sent to cell j thus cell j will not acquire channel r.

(b) If cell j was in borrowing update mode requesting for the same channel r, cell i will send it a reject RESPONSE message and cell i will not be able to acquire channel r.

(c) If cell j is in local mode, it includes channel r in its interfered channel set I_j and responds with a grant RESPONSE message. Hence, cell j will not be able acquire channel r until cell i sends a RELEASE(i, r) message to cell j.

Theorem 2 *The algorithm is deadlock free.*

Proof: We show that there is no circular wait in the algorithm due to the use of timestamps. If all cells in a given interference region are in local mode all requests can proceed in parallel and there is no deadlock. If a cell is requesting for a channel in borrowing update mode then it's request can be deferred by another cell in borrowing search mode if it's request has a lower timestamp. In this case the channel doing the search cannot be blocked by the cell doing the update since it has a higher timestamp than the search message. Thus the search proceeds and there is no deadlock. A search request from cell i can be deferred by another cell j which is carrying out a search or update with a lower timestamp. In this case, cell i will not defer cell j's request, thus there is no deadlock. The search request from cell i can also be deferred by a another cell l which is in local mode and it has a pending local request with a lower timestamp, i.e. $ts_i > ts_l$. This will happen if cell l has sent out its channel use set to another cell p trying to acquire a channel in search mode and has not received ACQUISTION message from it. Since, cell l's request was generated after sending response to cell p it implies that $ts_l > ts_p$. Therefore $ts_i > ts_l > ts_p$ which in turn implies that there can be no cyclic waiting. In all other cases no message is deferred, hence there can be no deadlock.

5. Performance Analysis and Comparisons

In this section we provide a preliminary estimate of the performance of the proposed adaptive channel allocation scheme in terms of channel acquisition time and message complexity; we also compare our scheme with some existing schemes based on search and update. We use the following notations.

N	Number of nodes in the interference region of any cell.
N_{borrow}	Average number of neighbors (of a cell) in the *Borrowing* mode.
N_{search}	Average number of cells (in the neighborhood of any cell) initiating a simultaneous search/update to acquire a channel.
α	Maximum number a cell attempts to borrow a channel using update scheme; after α attempts, it uses the search scheme.
m	Average number of attempts using update scheme; $m \leq \alpha$.
ξ_1	Fraction of channel acquisitions by a cell in the local mode.
ξ_2	Fraction of channel acquisitions by a cell in the borrowing mode using update scheme.
ξ_3	Fraction of channel acquisitions by a cell in the borrowing mode using search scheme ($\xi_1 + \xi_2 + \xi_3 = 1$).
n_p	Number of primary cells of a channel r which are in interference region of any cell c (used for Advance update scheme [4]).

We make the following observations:

- When a local channel is acquired, the cell has to exchange N_{borrow} ACQUISITION messages and N_{borrow} RELEASE messages. Thus, for a primary channel acquisition, the message complexity is $2N_{borrow}$, and the channel acquisition time is zero.

- In the *borrowing* mode, assume that a channel is acquired after m attempts using the update scheme. For each attempt, N REQUEST, N RESPONSE and N RELEASE messages need be exchanged. Hence the channel acquisition time is $2mT$ and the message complexity is $3mN$.

- If a channel cannot be acquired after α attempts using update, the cell goes into the search mode. Assuming that the current request timestamp is smaller than that of $N_{search}-1$ other simultaneous searches, the channel acquisition time is $(2\alpha + N_{search} + 1)T$ and the message complexity is $3\alpha N + 4N$ (in the search scheme a REQUEST, RESPONSE, ACQUISITION and RELEASE message need be sent to all neighbors in the interference region).

Combining the above observations we get the average channel acquisition time of our proposed adaptive scheme as

$$\{2m\xi_2 + (2\alpha + N_{search} + 1)\xi_3\}T$$

and the average message complexity as

$$2\xi_1 N_{borrow} + 3\xi_2 mN + \xi_3(3\alpha + 4)N$$

We can easily estimate the message complexity and the channel acquisition time for 3 distinct channel acquisition schemes available in the literature, e.g., the basic search scheme [4], the basic update scheme [4], and the advanced update scheme [3] and the results are summarized in the Table 1.

Algorithms	Message Complexity	Channel Acquisition
Basic Search	$2N$	$(N_{search}+1)T$
Basic Update	$2Nm+2N$	$2Tm$
Advanced Update	$(1-\xi_1)(2n_pm+n_p(m-1))+2N$	$(1-\xi_1)2Tm$
Adaptive (Proposed)	$2\xi_1 N_{borrow}+3\xi_3 mN+2\xi_3(\alpha+2)N$	$\{2m\xi_2+(2\alpha+N_{search}+1)\xi_3\}T$

Table 1. Comparison of Different schemes in General

In the *basic update* scheme, messages are exchanged between cells for each channel allocation, irrespective of the load. In the *adaptive* scheme, a cell in the *local* mode need not ask other cells before allocating a channel and needs to send the ACQUISITION message only to neighbors which are in the *borrowing* mode. Thus at uniformly low loads, no messages need be exchanged. The *adaptive* scheme selects a channel to borrow depending on the *borrowing criterion* which takes into consideration the number of borrowing neighbors a cell may have and hence reduces the chances of a collision. This way it saves on the number of attempts. When the load becomes high, the *update* scheme can suffer from starvation. In the *update* scheme there is always a finite probability of collision on every channel request and thus a cell can see unlimited delays. The *adaptive* scheme switches to *borrowing search* mode whenever the number of attempts to acquire a channel in borrowing mode exceeds a bound and hence provides fair service to each cell.
At a low system load, channels are acquired locally, i.e., $\xi_1 = 1$ and hence, $m = 0$, $N_{search} = 1$ and $N_{borrow} = 0$; performance metrics for different algorithms under such conditions are shown in Table 2. When the load varies, the maximum and minimum cost of each algorithm is shown in Table 3.

6. Conclusion

We have proposed an adaptive distributed channel allocation algorithm for use in wireless networks. The proposed scheme works well under all system loads; we observed the following:

- The scheme is optimal at uniformly low loads as all cells are in the *local* mode and no messaging is required. When the load is unbalanced with hot spot nodes surrounded by lightly loaded nodes, our algorithm behaves as the update scheme; the algorithm saves on the number of messages by sending ACQUISITION only to neighbors which are currently *borrowing*. By carrying out a heuristic search for the channel to borrow, chances of collision are further reduced. When system load becomes uniformly high, the *adaptive* scheme switches to searching and thus provides a bounded allocation time to all cells.
- The algorithm is deadlock free and avoids starvation.
- The message complexity is low and it's distributed nature makes it highly scalable.
- The algorithm provides fair service to all cells without compromising on any reuse issues.

Our proposed scheme also overcomes one problem associated with the advanced update scheme of [4]. The advanced update scheme saves on messages by sending request messages only to cells which are primary with respect to a cell ($NP(c,r)$), but this can result in no two cells in the same interference region obtaining a channel and unfairness. Consider cells c1 and c2 request for a channel which is a primary channel for p1 and p2 (see Figure 11). c1 has a lower timestamp but messages of c2 overtake those of c1 and reach p1, p2 earlier. Now both p1 and p2 send a *grant* message to c2, and a *conditional grant* to c1. Since c1 does not receive any *grant* message from $some\ p' \in NP(c,r) \cap IN(c')$, it's request fails; Thus c2 is able to acquire the channel, even though it has a larger timestamp. These scenarios are not possible in our scheme since the request is sent to all neighbors.

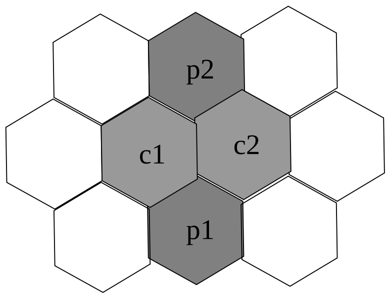

Figure 11. Advanced Update Scheme [4]

Our adaptive scheme also compares well with the advanced search scheme of [8] which uses the concept of the *Allocated* channels. The scheme of [8] adapts to load in that it allows a cell to keep a channel once it is allocated to it; thus, at transient high loads a cell can satisfy requests from its allocated set. However, when requests cannot be satisfied from the allocated set, the cell c requests for the allocated, busy and transfer sets of each cell in IN_c. If there are no free unallocated channels, additional three messages, TRANSFER(r), AGREE, KEEP(r) or RELEASE(r) are required to transfer a channel r from another cells allocated set to c; more than

Algorithms	Message Complexity	Channel Acquisition
Basic Search	$2N$	$2T$
Basic Update	$4N$	$2T$
Advanced Update	$2N$	0
Adaptive (Proposed)	0	0

Table 2. Comparison of Different Algorithms under Low Load

Algorithms	Message Complexity		Channel Acquisition	
	Minimum	Maximum	Minimum	Maximum
Basic Search	$2N$	$2N$	$2T$	$(N+1)T$
Basic Update	$2N$	∞	$2T$	∞
Advanced Update	N	∞	0	∞
Adaptive (Proposed)	0	$2\alpha N + 4N$	0	$(2\alpha N + 1)T$

Table 3. Bounds for Different Algorithms

one round of TRANSFER, AGREE and RELEASE messages may need to be exchanged by c, if there was a request to transfer r from more than one cell. In comparison, our adaptive scheme transfers channels with one round of messaging and the performance is equivalent to the basic search scheme under such load conditions.

References

[1] D. C. Cox and D. O. Reudink. Increasing channel occupancy in large scale mobile radio systems: dynamic channel reassignment. *IEEE Transactions on Vehicular Technology*, VT-22:218–222, Nov. 1973.

[2] S. Das, S. Sen, and R. Jayaram. A dynamic load balancing startegy for channel assignment using selective borrowing in cellular mobile environment. In *MOBICOM 96*, pages 73–84, Nov. 1996.

[3] X. Dong and T.-H. Lai. Dynamic carrier allocation strategies for mobile cellular networks. Technical Report OSU-CISRC-10/96-TR48, Dept of Comp and Info Science, Ohio State University, 1996.

[4] X. Dong and T.-H. Lai. Distributed dynamic carrier allocation in mobile cellular networks: Search vs. update. In *Proceedings of the 17th International Conference on Distributed Computer Systems*, pages 108–115, Baltimore, Maryland, May 1997.

[5] T. Imielinski and B. Badrinath. Wireless computing: Challenges in data management. *Communications of the ACM*, 37(10), Oct. 1994.

[6] I. Katzela and Naghshineh. Channel assignment schemes for cellular mobile telecommunication systems. *IEEE Personal Communications*, 3(3):10–31, June 1996.

[7] V. H. Macdonald. Advanced mobile phone service: the cellular concept. *Bell System Technical Journal*, 58:15–41, Jan. 1979.

[8] 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, 1995.

[9] J. Tajima and K. Imamura. A strategy for flexible channel assignment in mobile communication systems. *IEEE Transactions on Vehicular Technology*, VT-37, May 1988.

[10] M. Zhang and T. S. Yum. Comparisons of channel assignment strategies in cellular mobile telephone systems. *IEEE Transactions on Vehicular Technology*, 38, Nov. 1989.