Distributed List Coloring: How to Dynamically Allocate Frequencies to Mobile Base Stations*

Naveen Garg[†] Marina Papatriantafilou Philippas Tsigas
Max-Planck Institute für Informatik,
Im Stadtwald, 66123 Saarbrücken, Germany
email: {naveen|ptrianta|tsigas}@mpi-sb.mpg.de

Abstract

To avoid signal interference in mobile communication it is necessary that the channels used by base stations for broadcast communication within their cells are chosen so that the same channel is never concurrently used by two neighboring stations. We model this channel allocation problem as a *generalized list coloring problem* and we provide two distributed solutions, which are also able to cope with crash failures, by limiting the size of the network affected by a faulty station in terms of the distance from that station. Our first solution uses a powerful synchronization mechanism to achieve a response time that depends only on Δ, the maximum degree of the signal interference graph, and a failure locality of 4. Our second solution is a simple randomized solution in which each node can expect to pick $f/4\Delta$ colors where f is the size of the list at the node; the response time of this solution is a constant and the failure locality 1. Besides being efficient (their complexity measures involve only small constants), the protocols presented in this work are simple and easy to apply in practice, provided the existence of distributed infrastructure in networks that are in use.

*This work will also be published elsewhere.
†Partially supported by the EU ESPRIT LTR Project No. 20244 (ALCOM-IT)

1 Introduction

The integration of mobile communication and computing in fixed networks introduces new issues and problems in distributed computing. The ability of a host to be reachable while moving requires some wireless form of communication. A wireless medium intrinsically supports broadcast communication within a specific region, called a *cell*. To provide communication between different cells, a specific host, called *base station*, is associated with each cell. The set of base stations are interconnected via a fixed network, which usually does not physically support broadcast communication, but rather point-to-point message transmission.

New problems and issues which are introduced due to the mobility of hosts include: bandwidth management and frequency allocation, naming and addressing mobile hosts, logical network structure maintenance and routing, locating the mobile hosts, file system organization, data management, consistency guarantees of multicasting under host migration, coping with the less secure and less reliable environment of wireless and broadcast communication, coping with resource (energy) constraints of mobile hosts and, of course, definition of appropriate performance measures for evaluating solutions.

Here we address the problem of frequency (or channel) allocation to base stations. As mentioned before, a mobile host can establish communication with other entities of the network only through the base station associated with the cell in which it is present. In order to satisfy a communication request of a mobile host, the respective base station should allocate, depending on the bandwidth required, a certain number of wireless channels to it. Wireless channels are divisions of the frequency spectrum provided for broadcast communication; the signal carried on a wireless channel is carried on the respective frequency. Henceforth, the terms "channel" and "frequency" are used interchangeably. The wireless channels should be chosen by the base stations in such a way that no interference between signals can occur. A solution to the frequency allocation problem should guarantee that if a channel is used for a communication session of a mobile host in a specific cell, it should not be used concurrently for other communication sessions, except in distant cells, outside the scope of the signal.

In this paper we model the interference between signals as an *interference graph*, in which the nodes are the base stations (or, equivalently the cells) and the edges between them represent possible signal interferences in case of concurrent use of the same channel. The *free frequencies* at a base station are the frequencies of the spectrum that are not in use by the base station or any of its neighbors in the interference graph. Thus this is the set of frequencies from which channels have to be allocated to satisfy the requests at this base station. Note that the set of free frequencies for a base station changes over time. Furthermore, the same frequency could potentially belong to the set of free frequencies of two adjacent base stations and hence even when a base station is picking channels from its set of free frequencies it needs to ensure that there is no interference. The problem of allocating frequencies to mobile hosts by the base stations can now be viewed and analyzed as a generalized version of the *list coloring* problem. In the list coloring problem, every vertex of the graph has associated with it a list of colors which in our case are the set of free frequencies. The requirement is to find a proper vertex-coloring of the graph such that each vertex is colored with one of the colors in its list. The list coloring problem was introduced in [9]; sequential protocols for solving it can be found in [1, 14, 18]. The relation between the classical list coloring and the channel allocation problems has also been noticed in [15]. In the generalized version defined in this paper, each vertex needs to be colored with a certain number of colors from its list—the number of colors required for a vertex being the number of requests at the base station. Thus any protocol for the list coloring problem that is *long-lived*, i.e. it can be invoked multiple times, with lists and requests that vary in time, can be used for solving the channel allocation problem in a dynamic manner.

A solution to the channel allocation problem should aim at minimizing the connection setup time, i.e. the time when a communication request is issued to the time the session is established. It must also aim at minimizing the amount of information (number and size of messages) that needs to be exchanged per request. A solution should also aim at maximizing the number of satisfiable requests; clearly this implies that it should adapt fast to temporal variations in channel demand in different cells. Another evaluation criterion of a solution is the number of *hand-offs*; hand-off is a situation when some frequency used by a mobile host for a communication session has to change during the session, in which case the communication is interrupted until a new frequency is found. For data communication the problem is solved by buffering, but the quality of on-line communication (e.g. voice) can degrade if the number of hand-offs and the time it takes to serve them are high. For this reason, apart from the inter-cell hand-offs—which occur when a mobile host moves from one cell to another— a solution must avoid introducing intra-cell hand-offs (i.e within a cell). In addition to the performance evaluation measures, a very important property of a solution is its ability to cope with failures, since they are a fact of life in any distributed setting. A fault-tolerant solution should guarantee that what happens in a specific neighborhood is not reflected in the whole network. One criterion to measure this property is the *failure locality* [7] which measures the size of the network affected by a faulty station in terms of the distance from that station. Clearly, the optimization criterion here is to minimize the failure locality of a solution. To make efficient use of the bandwidth available we also need to ensure that the number of free frequencies needed at a base station so that its requests can be satisfied is small. The work presented here not only improves with respect to efficiency and fault-tolerance over previous solutions to the channel allocation problem, but also gives a new perspective to study the problem.

1.1 Results and Previous Work

The first solutions to the channel allocation problem were static, in the sense that each base station was assigned a fixed set of frequencies that it could use; this strategy clearly does not behave well under temporal variations in channel demand. Existing networks for cellular telephony employ dynamic, but centralized solutions: there exists a *channel manager* (called Mobile Telephone Switching Office or MTSO in the cellular telephony terminology), which is a central fixed base station that collects the requests from the base stations and decides how to satisfy them (for a brief description of the system see [12, 13]). Hybrid settings, in which some base stations are allocated frequencies in a static manner, while others are allocated frequencies dynamically (but centrally), have also been considered and studied [15, 17].

Centralized solutions have the drawback that they imply dependency on a central node. This, in turn, implies poor fault-tolerance; whatever happens to the central station is reflected in the whole network. For the network to be able to handle rapidly changing load patterns it is important that all the base stations inform the channel manager every time a frequency is requested or released. This however would lead to significant traffic. On the other hand any grouping of frequency requests and releases by the base stations could lead to large waiting times for the mobile hosts. While this trade-off between the amount of network traffic generated and the response time that the solution offers to the mobile hosts is inherent to any solution for this problem it is all the more significant in centralized solutions where each node has to communicate with a central node some distance away in the network. A distributed solution on the other hand has better fault tolerance. Additionally, since the nodes communicate only with their neighbors, the amount of network traffic generated is much less and the trade-off mentioned above is no more a serious consideration. Furthermore, a distributed solution is better poised to exploiting and preserving the locality inherent in the problem.

There have been attempts to solve the problem in a distributed manner. In [8] the proposed protocol involves participation of the mobile hosts themselves in the procedure of frequency assignment; this is not desirable, because of energy constraints. In [16] the problem is analyzed as a multiple mutual exclusion problem. In order to satisfy requests, a base station requires one channel at a time. The connection set-up time is measured as the number of rounds of message-exchanges with the neighboring base stations and can be as high as the size of the spectrum. The size of messages too can be as high as the size of the frequency spectrum. The protocol for allocating and releasing frequencies might lead to a situation in which a busy cell which has collected a large set of frequencies prevents its neighbors from satisfying their own requests thus starving these base stations. Furthermore the solution in [16] uses time-stamps as a priority scheme to achieve exclusion; this can result in arbitrary long process waiting chains, which implies a failure locality equal to the diameter of the network. Thus the failure of some station could potentially cause the whole network to fail.

Since neighboring cells cannot be using the same frequencies concurrently, the frequencies can be viewed as resources and the channel allocation problem as one of resource allocation. However, this view ignores a crucial aspect of the problem – to satisfy the requests in a cell, the base station does not need specific frequencies but only a certain number of them. Thus if viewed as a resource allocation problem we might have a situation in which two adjacent base stations decide to pick the same frequency and so one of them has to wait for the other to release the frequency thereby resulting in response times that depend on the duration that the frequencies are in use. In this paper we view the problem as a generalized list coloring problem and we first show a way to solve it by deriving the long-lived generalized distributed equivalent of a known sequential solution to the problem [1]. We do this by using the double doorway and fork release synchronization mechanisms introduced in [7]. With respect to request satisfiability, the sequential solution would satisfy the requests in every cell, given that for every node, the sum of the number of its requests and of those of its outgoing-edge-neighbors in an acyclic orientation of the graph, does not exceed the number of free frequencies. For our distributed protocol the requests in a cell are satisfied provided the sum of its requests and those of its neighbors is at least as large as the number of free frequencies; the exact condition depends on certain timing and synchronization aspects and is detailed in Section 4; The number of messages exchanged and the response time for a request are $O(\Delta^2)$ for general interference graphs and $O(\Delta)$ if the interference graph is planar—a natural setting for mobile communication networks— where Δ is the maximum degree of the graph and is independent of the size of the graph in this setting. Only a number Δ of the messages exchanged have size that depends on the number of frequencies requested; the rest of the messages need only carry 2 bits of information. The failure locality of the solution is 4; in asynchronous networks any deterministic protocol has a failure locality of at least 2 [7]. Our second solution for the dynamic channel allocation problem employs randomization and requires constant response time and $O(\Delta)$ messages, in order to satisfy $f/4\Delta$ requests (in an expected sense), where f is the number of free frequencies. The failure locality of the randomized solution is only 1, which is also the lower bound for randomized protocols. Besides, it should be mentioned that, as stated in the requirements from a good solution, our protocols do not introduce any unnecessary (inter-cell) hand-offs.

2 Preliminaries

2.1 Computation Model and Problem Statement

As mentioned in the introduction, we model the problem of channel allocation as a generalization of the list coloring problem. The network is described with an interference graph $G = (V, E)$. Each node $v_i \in V(G)$ $(1 \leq i \leq n)$, also called *process*, models a base station of the network. There is an edge between two nodes if there is a signal interference when the respective base stations broadcast on the same frequency. The maximum degree Δ of G is expected to be independent of the number of nodes n. Each node v_i has an associated set of colors L_i, which models the set of free frequencies from which the frequencies to be allocated to v_i must be chosen. Finally, each node v_i has an associated number r_i of color (channel) requests.

Each process v_i can communicate with its neighbors in G using messages. No assumption is made on the relative speeds of the stations or on the communication links connecting them, i.e. the network is considered to be completely asynchronous. For well designed wide area networks the interference graph G is planar. However our solutions do not assume planarity and are therefore appropriate for all types of networks, including nano-cellular ones (e.g. for multi-story office buildings).

The requirement is to *properly* color each node v_i of G, i.e. that each v_i gets r_i colors from its list L_i, in a way that none of the colors used for v_i is used for any of its neighbors in G. This is a generalization of the list coloring problem, which requires that each v_i is properly colored with *one* color from its list L_i. The *list chromatic number* $\chi_l(G)$ of G is the smallest number k for which, for any assignment of a list L_i of size at least k to every vertex $v_i \in V(G)$ it is possible to color G so that every vertex gets one color from its list. If $\chi_l(G) \leq k$ the G is said to be *k-choosable*.

We measure the efficiency of a solution with respect to request satisfiability as the size of the list which is sufficient to ensure that the node gets properly colored. The time it takes for a request to be satisfied is measured in units of the maximum message transmission delay in the network; it should be pointed out that this bound is not assumed for the sake of the correctness of any of our protocols, but only for the sake of measuring the response times. It is required that any solution is *starvation free*, that is, as long as a process does not fail the response time for its requests is bounded. Failure locality is introduced in order to measure the effect of process stopping failures. An algorithm has failure locality m if a process is free from starvation even if some process outside its m-neighborhood has failed by stopping. (The *m-neighborhood* of a process is the set of processes within distance m from it.)

2.2 Sequential Solutions for List Coloring

The problem of list coloring a graph generalizes the vertex coloring problem and is hence **NP**-complete [5]. Given a graph of maximum degree Δ, one straightforward way of coloring it with $\Delta + 1$ colors is to consider vertices one at a time and assign them a color different from their neighbors. Clearly, we need only $\Delta+1$ colors to ensure that for each vertex we can find a color different from its neighbors. This simple strategy can be adapted to list-coloring – for each vertex we pick a color from its list that is different from the colors of its neighbors. Therefore, if each list is of size at least $\Delta + 1$ we can always list-color the graph. Note that if the graph was complete we would actually require that the lists be of size $\Delta + 1$.

If we have an initial acyclic orientation of the edges of the graph then we could first color all vertices that are sinks (no two sinks are adjacent) by picking an arbitrary color from their lists. We then remove these sinks and color the new sinks created taking care to assign them a color different from their neighbors. It is now easy to see that a list coloring is always possible if each vertex has a list of size more than its out-degree in the initial orientation. Alon and Tarsi [1] use linear algebraic techniques to argue that list of these sizes are sufficient even when the initial orientation of the edges is not acyclic provided the orientation satisfies some properties with respect to the number of Eulerian subgraphs.

An elegant argument due to Thomassen [18] shows that every planar graph is 5-choosable. In particular, a planar graph is list colorable if every vertex in the graph not on the infinite face has a list of size 5 while vertices on the infinite face need only have lists of size 3. This is optimal since there are planar graphs known which are not 4-choosable [19].

Note that all these solutions for the list coloring problem are strictly sequential. So while they are not interesting from the distributed view-point, they can be used to provide centralized solutions to the dynamic channel allocation problem as follows. The channel manager maintains the set of frequencies that are in use at each base station. Every time the channel manager receives requests for frequencies from some base stations it runs one of the sequential algorithms mentioned above to allocate the frequencies.

3 Deterministic Distributed Approach

We show how we can derive the generalized distributed equivalent of the solution of the

```
var Busy_i, free_i: sets of busy, free frequencies
    ∀j : v_j ∈ N(v_i), Occupied_ij: v_i's knowledge of Busy_j
    ∀j : v_j ∈ N(v_i), l_ij: last synchronization message received from v_j
    ∀j : v_j ∈ N(v_i), pr_ij: privilege (fork) shared between v_i and v_j—one of them holds it at a time
param p_i: priority number (from the initial color assignment) in {1, ..., C}

procedure acquire(r_i);
    1. ∀j : v_j ∈ N(i), wait until l_ij ≠ sync_1 ; {1st doorway: wait until true for each v_j }
       send sync_1 to all neighbors ;
    2. wait until ∀j : v_j ∈ N(i), l_ij ≠ sync_2 ; {2nd doorway: wait until true for all v_j's }
       send sync_2 to all neighbors ;
    3. request and wait for all privileges pr_ij;          { smaller p_i has higher priority }
    4. choose arbitrarily a set S of min(|Free_i|, r_i) frequencies from Free_i ;
    5. send conf(S) to all neighbors and wait for ack's ;
    6. Busy_i ← Busy_i ∪ S ; Free_i ← Free_i − S ;
    7. send sync_3 to all neighbors ;                       {signal for doorways }
end
procedure release(f);
    1. Busy_i ← Busy_i − f ; Free_i ← Free_i ∪ f ;
    2. Send rel(f) to all neighbors ;
end

on receiving sync_x from v_j do
   l_ij ← sync_x

on receiving conf(S) from v_j do
   for all frequencies f ∈ S do
      Free_i ← Free_i − f; Occupied_ij ← Occupied_ij ∪ f ;

on receiving rel(f) from v_j do
   Occupied_ij ← Occupied_ij − f ;
   if ∀j : v_j ∈ N(v_i), f ∉ Occupied_ij  then Free_i ← Free_i ∪ f;
```

Figure 1: Protocol DET-DLC for node v_i

previous paragraph that list colors a graph given an initial acyclic orientation.

An acyclic orientation can be obtained by first executing a distributed vertex coloring protocol and then orienting each edge as outgoing from the higher color endpoint. Arbitrary graphs can be colored in a distributed manner with $\Delta + 1$ colors [2], while planar with only 6 [10]. Starting from such an acyclic orientation, let the sinks concurrently and independently list-color themselves first. There is no conflict between them when they choose colors since no two sinks can be neighbors and each node v_i knows the colors of each node v_j in its immediate neighborhood, denoted by $N(v_i)$. Then let them communicate to their neighbors the colors they chose and reverse their incident edges. The resulting orientation is again acyclic, so the new sinks can list-color themselves next. By repeating this procedure, in time equal to the size of the longest *chain* (directed path) in the initial orientation, all the nodes of the graph will be list-colored, provided that for each node, the sum of the number of its requests and of those of its outgoing-edge neighbors of the initial orientation does not exceed its list size. Note that in the initial orientations assumed, the longest chains are $\Delta + 1$ and 6 for general and for planar graphs, respectively. This scheme not only works for the case when each node is interested in choosing colors once (one-shot protocol), but it can also serve as a long-lived solution. The result is a token-passing-like protocol [4, 6]; any process that becomes a sink at some point holds the token, which is a privilege to choose frequencies if there are pending requests. However, there is a serious drawback. The worst-case for the response time for a request equals the size of the longest chain in some orientation in an execution. The chains of the initial orientation get modified by becoming longer at their "tails" and shorter at their "heads" due to the edge-reversal mechanism. Although in this manner in a synchronous system the longest chain size does not grow, if the system is asynchronous, due to the speed difference of the processes, these modifications may even cause chains of length n.

Therefore, for the long lived protocol we should use a different approach. The synchronization mechanism of Choy and Singh [7], which was presented for the dining philosophers problem, is shown to be appropriate —for ease of reference we briefly rephrase the principles of that synchronization mechanism together with the description of our final solution (cf. fig. 1). An initial vertex coloring of the graph is assumed, in order to serve as a priority scheme. Assume C colors are used. Between neighboring nodes that request frequencies at the same time, the one with smaller color (p_i) chooses first. In order to avoid starvation of the nodes with high color due to preemption by their lower colored neighbors, a double doorway synchronization is used. The first doorway is equivalent to the process asking for permission to enter and waiting until it receives one (a message $\neq \mathtt{sync}_1$) from *each* one of its neighbors. Although this alone could guarantee starvation freedom — provided that a process that has already crossed the doorway defers giving permission to its neighbors until after it has chosen frequencies — it could result in exponential response time [7] due to processes' asynchronously crossing the doorway. So, after having crossed the first doorway (sending $\mathtt{sync}_1$ messages to all its neighbors) a node has to wait for crossing the second one, which is there to mend the above mentioned deficiency, by synchronizing the crossing times—it is equivalent to the process waiting until it finds an instance that none of its neighbors is past this second doorway. After having crossed both doorways (sending $sync_2$ messages to all neighbors) a process v_i has to wait for its neighbors $v_j \in N(v_i)$ that are also past both doorways, to pass to it the pr_{ij} privileges. The way that privileges are requested and passed from one node to the other is such that the maximum length of any process waiting chain is bounded by four at any time instant in any execution [7]: A node first requests the privileges from its higher priority neighbors and then from the lower priority ones. A competing node v_i, which has crossed both doorways, gives the privilege to a lower priority neighbor only of it has not collected the privileges from all its higher priority neighbors; otherwise it defers answering until after it has finished choosing frequencies. On the other hand, v_i gives the privilege to a higher priority neighbor if it has not collected all the other privileges.

In order that the neighbors of a process v_i have a correct view of their free frequency sets when they are choosing, v_i informs them about the frequencies it picked using `conf` messages and waits for acknowledgments before it signals them with $\texttt{sync}_3$ messages to make their own choices. When a frequency is not used any more, v_i informs its neighbors using `rel` messages to update their sets.

Regarding the bandwidth use, the following lemma holds:

Lemma 3.1 *In any execution of protocol DET-DLC, after a process v_i has executed step 1 of procedure acquire, it is guaranteed to get r_i frequencies if*

$$|\texttt{Free}_i| \geq r_i + \sum_{v_j \in N(v_i)} r_j$$

at that point in the execution.

Proof. (Sketch) Since a node, after having crossed the first doorway, will have to wait in the worst case for all its neighbors to select frequencies, but not more than once, the lemma follows. □

Theorem 3.1 *Protocol DET-DLC is a correct distributed solution for the channel allocation problem; it does not introduce intra-cell hand-offs and has failure locality 4. The number of messages exchanged and the response time for satisfying r requests are $O(\Delta^2)$ ($O(\Delta)$ if G is planar). The size of the messages is 2 bits, except from Δ messages which are $O(r)$ bits.*

Proof. (Sketch) The correctness of the protocol follows from the following argument: Assuming FIFO channels, node v_i's `rel` messages arrive before its next `conf` messages. This guarantees consistent views of the free frequencies by neighboring nodes. Combining this with the fact that no two neighboring processes choose frequencies at the same time (which follows from [7]) it follows that no frequency is used concurrently by neighboring processes.

The fault tolerance of the protocol results from the fault tolerance of the synchronization structure used. The processes waiting chains that are formed, which are originally of length $C + 2$, are truncated to only 4 (1 for each doorway + at most 2 inside the second doorway) using the privilege release mechanism described; therefore, the failure locality of the solution is 4.

The waiting time of a process after having crossed both doorways is $O(C)$— due to the privilege transfer mechanism from higher color nodes to the lower colored ones. In order to cross the second doorway, which is synchronous, it might have to wait—once and no more— for each one of its neighbors to cross it, complete and send the $\texttt{sync}_3$ messages, therefore, the waiting time there is $O(\Delta C)$. In order to cross the first doorway, which is asynchronous, a process in the worst case has to wait for the slowest of its neighbors that had already crossed it before, to finish and and send the $\texttt{sync}_3$ messages. Therefore, the total response time is $O(\Delta C)$, which, for general graphs is $O(\Delta^2)$ and for planar ones $O(\Delta)$. The size of the messages exchanged for the synchronization phase need be only 2 bits; only the `conf` messages need be $O(r)$ bits, to describe the frequencies picked. □

```
var Busy_i, Free_i: sets of busy, free frequencies
    ∀j : v_j ∈ N(v_i), Occupied_ij: v_i's knowledge of Busy_j


procedure acquire(Busy_i,Free_i);
    1. Pick, randomly, a set S of ε · |Free_i| frequencies from Free_i;
       Busy_i ← Busy_i ∪ S ; Free_i ← Free_i − S ;
    2. Send pick(S) to all neighbors and wait for replies ;
    3. for all frequencies f ∈ S do
           if some neighbor has replied 'No' for f then
              Busy_i ← Busy_i − f ; Free_i ← Free_i ∪ f ; S ← S − f ;
    4. Send conf(S) to all neighbors ;          { S is now the set of acquired frequencies }
end
procedure release(f,Busy_i,Free_i);
    1. Busy_i ← Busy_i − f ; Free_i ← Free_i ∪ f ;
    2. Send rel(f) to all neighbors ;
end

on receiving pick(S) do
   for all frequencies f ∈ S do
       if f ∈ Busy_i then Reply 'No' for f else Reply 'Yes' for f ;

on receiving conf(S) from v_j do
   for all frequencies f ∈ S do
       Free_i ← Free_i − f; Occupied_ij ← Occupied_ij ∪ f ;

on receiving rel(f) from v_j do
   Occupied_ij ← Occupied_ij − f ;
   if ∀j, f ∉ Occupied_ij then Free_i ← Free_i ∪ f;
```

Figure 2: Protocol RAND-DLC for node v_i

4 Randomized Distributed Approach

Our previous solution for the distributed list coloring problem relied on mutual exclusion. This approach allowed us to color a node provided the total number of colors needed by a node and its neighbors did not exceed the size of the list at the node. As observed before, this is a necessary condition for list coloring if the graph has no special structure. Furthermore, it seems that the only way to obtain a list coloring without relaxing this condition is by a mutual exclusion approach.

Suppose instead that the list at a node was large, much larger than the total number of colors needed by the node. Could we do better? In this section we present a randomized algorithm that assigns colors to a node such that they do not conflict with the neighbors; the number of colors assigned depends (with high probability) on the ratio of the size of the list and the degree.

Let node v_i have degree Δ and a list L_i of cardinality f. Our protocol for picking the colors is to pick randomly, an ϵ fraction of the colors in L_i; thus each color in the list is picked with a probability ϵ. The node then checks with its neighbors to ensure that some color it picked is not picked by a neighbor; the color is dropped if such is the case. Thus a color picked by two neighbors could potentially be dropped by both of them. Node v_i, as in protocol DET-DLC, maintains two sets $\texttt{Free}_i$, the set of frequencies that are available to be picked and $\texttt{Busy}_i$, the set of frequencies that are in use by node i; it also maintains for each neighbor v_j a set $\texttt{Occupied}_{ij}$, its view of the set $\texttt{Busy}_j$. Our protocol ensures that at any point a frequency that is in the `Free` set of v_i does not belong to its `Busy` set or to the `Busy` set of any of its neighbors. Thus, after node v_i picks $\epsilon \cdot |\texttt{Free}_i|$ frequencies from $\texttt{Free}_i$ it updates the sets $\texttt{Busy}_i$ and $\texttt{Free}_i$, adding these frequencies to $\texttt{Busy}_i$ and deleting them from $\texttt{Free}_i$. These updates are tentative since not all the frequencies picked will be eventually acquired. We also require that these updates are made without interruption; the node suspends all message handling while making these changes to the sets. Node i then sends the list of frequencies picked to its neighbors to check if they are in conflict. After it has received replies from all its neighbors it knows whether a particular frequency can be acquired, in which case this frequency is left in the `Busy` set, else it is deleted from the `Busy` set and added back to the `Free` set. Node v_j, on receiving the list of picked frequencies from its neighbor v_i, checks, for each frequency in this list, whether it belongs to its `Busy` set. Only if the frequency does not belong to $\texttt{Busy}_j$ does it reply with a 'Yes', saying 'No' otherwise. As in DET-DLC, when v_i acquires or no longer uses a frequency it informs its neighbors using `conf` and `rel` messages, respectively, in order to update their sets. When v_j receives the list of acquired frequencies from v_i it deletes all frequencies in this list from its `Free` set.

Theorem 4.1 *Protocol RAND-DLC is a correct distributed solution for the channel allocation problem; it does not introduce intra-cell hand-offs and has failure locality 1. With the exchange of 3Δ messages in 3 rounds of communication a node v_i gets at least $r = \mu - \sqrt{2\mu}$ frequencies with probability at least $1 - e^{-1}$, where $\mu = \frac{f}{4\Delta}$ and f is the number of free frequencies for v_i at that point of the execution; 2Δ of the messages are $O(r)$ bits long, while Δ of them are 1 bit long.*

Proof. (Sketch) For correctness, we need to argue that no two adjacent nodes can acquire the same frequency. Note that a frequency that is acquired by v_i is in $\texttt{Busy}_i$ from the point that it is picked by v_i to the point it is released. Node v_i acquires a frequency only after it receives replies from all its neighbors saying that the frequency is not in their `Busy` sets. Hence v_i cannot acquire the frequency if one of its neighbors has already picked it.

To compute the probability that a color that was picked by a node is retained, observe that the probability that a neighbor v_j also picked this color is ϵ. Since the color is retained when none of the neighbors picks it and since the number of neighbors who could be picking colors is at most Δ, the probability that a color is retained is at least $(1-\epsilon)^{\Delta}$.

Let X_j be a random variable that is 1 if the j^{th} color in the list (of some v_i) is acquired (and is 0 otherwise). Then the probability that $X_j = 1$ is at least $\epsilon(1-\epsilon)^{\Delta}$. Let further

$X = X_1 + X_2 + \ldots + X_f$ be the sum of these random variables. The expected value of the random variable X is at least $f \cdot \epsilon(1-\epsilon)^{\Delta}$ which is maximized when $\epsilon = 1/(\Delta+1)$. For this choice of ϵ, the expected number of colors acquired is at least

$$\mu = \frac{f}{\Delta}\left(1 - \frac{1}{\Delta+1}\right)^{\Delta+1}$$

Note that $\Delta \geq 1$ and hence the expected number of colors acquired is at least $f/4\Delta$. Thus with this fairly simple randomized scheme each node can expect to acquire a fraction $1/4\Delta$ of the free frequencies available to it. This compares very well with the best centralized algorithm where it is essential that a node have at least $\Delta + 1$ free frequencies for it to be able to acquire a frequency.

Since X is the sum of independent Bernoulli trials we can use Chernoff's [11] bounds to bound the probability that the number of colors acquired is at least $(1-\delta)\mu$ as

$$\Pr(X > (1-\delta)\mu) > 1 - e^{-\mu\delta^2/2}$$

Thus for $\delta = \sqrt{\frac{2}{\mu}}$ the probability that we acquire $\mu - \sqrt{2\mu}$ colors is at least $1 - e^{-1}$.

The failure locality of the solution follows from the fact that a node may be kept waiting only due to his neighbors (no answers are deferred), hence, the maximum waiting chain is of length only one. □

5 Discussion

We have shown two distributed protocols that solve the dynamic channel (frequency) allocation problem for networks of base stations that support mobile communication. By being distributed, they can take advantage of the locality and node independence in the network; thus they have good scalability properties, as opposed to currently existing centralized solutions, whose performance depends on the distances between the base stations and the respective channel managers. Another big advantage of the protocols presented is that they can tolerate node failures by limiting their effects to only a small neighborhood around the stations where they happen, and not letting them expand and affect the whole network. Besides being efficient (their complexity measures involve only small constants), the protocols are simple and easy to apply in practice, provided the existence of distributed infrastructure in networks that are in use.

In both the solutions presented, hand-offs may arise only when a mobile host changes cells. Then the communication must be interrupted and frequencies available in the new cell must be assigned to that session. For data communication the gap can be closed using buffering of the data packets until the new frequencies are allocated. For on-line communication, though, this cannot apply; hence, it is very important that the procedure be fast, since otherwise the quality of communication might degrade. Our protocols guarantee fast response to new communication requests (the first depending only on the number of neighbors and the second being a small constant), but they can be further enhanced in two ways: (i) by distinguishing the type of requests, regular and those that result from hand-offs, and servicing the latter with higher priority, and/or (ii) by keeping at every station a set of "back-up" frequencies that will be used to serve requests due to hand-offs, to ensure availability. The latter may restrict the bandwidth available for regular requests, but will preserve the quality of communication in presence of hand-offs. Furthermore, the same idea can be applied to ensure frequency availability and to save message exchanges for regular requests, as well. An interesting open problem is to analyze this trade-off between availability and bandwidth utilization.

Regarding the bandwidth utilization, our first protocol allows a node to satisfy its requests provided that the total number of its own and its neighbors' requests does not exceed the size of free spectrum at that node. This condition is necessary when the graph has no special structure. Is seems that the only way to obtain a solution without relaxing this condition is by an exclusion approach. Our second protocol employs randomization and allows a node to satisfy a number of requests that depends on the ratio of the free spectrum and the degree of the node. It also seems that this is the best bandwidth usage that a distributed protocol can achieve. It is an important open problem to either prove the two conjectures or design a distributed protocol which will preserves the first condition without employing exclusion.

References

[1] NOGA ALON AND M. TARSI Colorings and Orientations of Graphs. *Combinatorica* 12 (2) (1992), pp. 125–134.

[2] BARUCH AWERBUCH, ANDREW GOLDBERG, MICHAEL LUBY AND SERGE PLOTKIN Network Decomposition and Locality in Distributed Computation. In *Proceedings of FOCS 1989*, pp. 364–369.

[3] B.R. BADRINATH, ARUP ACHARYA AND TOMASZ IMIELINSKI Impact of Mobility on Distributed Computations. Operating Systems Review, Vol. 27, No. 2, Apr. 1993.

[4] VALMIR BARBOSA AND ELI GAFNI Concurrency in Heavily Loaded Neighborhood-Constrained Systems. *ACM Transactions on Programming Languages and Systems*, Vol. 11, No. 4, October 1989, pp. 562-584.

[5] M. BIRÓ, M. HUJTER AND ZS. TUZA Precoloring Extensions. I. Interval Graphs. *Discrete Math.* **100** (1992), pp. 267–279.

[6] K.M. CHANDY AND J. MISRA The Drinking Philosophers Problem. *ACM Transactions on Programming Languages and Systems*, Vol. 6, No. 4, October 1984, pp. 632-646.

[7] MANHOI CHOY AND AMBUJ SINGH Efficient Fault Tolerant Algorithms for Resource Allocation in Distributed Systems. *ACM TOPLAS*, Vol. 17, No. 3, May 1995, pp. 535-559.

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

[9] P.ERDÖS. A.L. RUBIN AND H. TAYLOR Choosability in Graphs. In *Proceedings of the West Coast Conference on Combinatorics, Graph Theory and Computing*, (Congr. Num 26), pp. 125–157, 1979.

[10] SUKUMAR GHOSH AND MEHMET HAKAN KARAATA A self-stabilizing algorithm for coloring planar graphs. *Distributed Computing*, (1993) 7:55-59.

[11] TORBEN HAGERUP AND CHRISTINE RÜB A Guided Tour of Chernoff Bounds. *Information Processing Letters* 33 (1989/90), pp. 305-308.

[12] STEFAN G. HILD A Brief History of Mobile Telephony *Technical Report No. 372, University of Cambridge, Computer Laboratory*, January 1995.

[13] G. IOANNIDIS Mobile Computing. *PhD Thesis, Columbia University*, 1992.

[14] TOMMY R. JENSEN AND BJARNE TOFT Graph Coloring Problems. *Wiley-Interscience Series in Discrete Mathematics and Optimization*, 1995.

[15] E. Malesinsca An Optimization Method for the Channel Assignment in Mixed Environments. In *Proceedings of ACM MobiCom '95*, to appear. Also available as TR. No. 458/1995, Fachbereich Mathematik, Technische Universität Berlin.

[16] Ravi Prakash, Niranjan G. Shivarati and Mukesh Singhal Distributed Dynamic Channel Allocation for Mobile Computing. In *Proceedings of PODC 1995*, pp. 47–56.

[17] Hans-Ulrich Simon The Analysis of Dynamic and Hybrid Channel Assignment. *Technical Report, Universität des Saarlandes*, SFB144–B1, 10/1988.

[18] Carsten Thomassen Every Planar Graph Is 5-Choosable. *Journal of Combinatorial Theory*, Series B 62, 180-181 (1994).

[19] M. Voigt List colorings of planar graphs. *Discrete Mathematics* **120**, pp. 215-219, 1993.