A Quorum-based Distributed Channel Allocation Algorithm for Mobile Systems

Chakarat Skawratananond and Vijay K. Garg*
Parallel and Distributed Systems Laboratory
Electrical and Computer Engineering Department
The University of Texas at Austin,
Austin, Texas 78712
{skawrata,garg}@ece.utexas.edu

Abstract - Since radio spectrum is a scarce resource, efficient allocation of frequency channels is critical for the performance of mobile systems. The *update approach* is a way to allocate radio channels among cells in distributed fashion. In update-based algorithms, each cell maintains its local knowledge about channels available for its use by exchanging messages among cells in its interference neighborhood. The existing update algorithms suffer from high message complexity or high storage overhead. In this paper, we present a distributed update-based algorithm that imposes lower message complexity, while requiring smaller storage overhead than existing algorithms.

I. Introduction

Since radio spectrum is a scarce resource in mobile systems, an efficient reuse of the radio spectrum allocated to the system is required as the population of mobile users continue to grow at the tremendous rate. Availability of radio channels, therefore, plays an important role in achieving good system performance. A channel can be reused in a spatially disjoint cell if *reuse constraints* are satisfied. Reuse constraints are conditions used to determine when a channel can be reused by other cells without causing an interference called *co-channel interference*. The reuse constraint we consider is the *minimum channel reuse distance,* D_{min}. Two cells can use the same channel without interfering each other if the distance between them is at least D_{min}.

To avoid co-channel interference each cell must [4] (1) compute the set of available channels, (2) select one channel from the set of available channels, and (3) acquire the selected channel. The procedure that performs task (1) and (3) is referred to as the *channel acquisition algorithm*, and the one that performs task (2) referred to as the *channel selection strategy*. Channel selection strategies have been extensively studied in the context of cellular telephone systems. In this paper, we present an efficient approach to accomplish task (1) and (3).

There are many ways to perform task (1) and (3). One simple approach is to have each cell requests a channel from a central controller who will ensure that the co-channel interference does not occur. This centralized approach does not scale well, and it has a single point of failure. Another approach is to distribute tasks and responsibilities to each MSS in the system. In this distributed approach, each MSS exchanges information with its neighbors within the co-channel inteference range so that it can make a decision about channel allocation in its cell based on its local knowledge. Therefore, distributed algorithm is more robust and scalable.

A distributed channel allocation algorithm should aim at minimizing the channel acquisition delay.[1] It must also aim at minimizing the amount of information (size and number of messages) that needs to be exchanged per request. Also, the algorithm must be able to cope with failures of the components of the network. One criterion is the *failure number* which measures the number of cells affected by a faulty support station. Clearly, the goal is to minimize the failure number of the algorithm.

We present a distributed algorithm in which each MSS bears an equal amount of responsibility for channel allocation control. Each cell is required to communicate only with a small subset of its neighbors within the co-channel inteference range in order to acquire a channel. No responses are ever deferred,

*supported in part by the NSF Grants ECS-9414780, CCR-9520540, a TRW faculty assistantship award, a General Motors Fellowship, and an IBM grant.

[1] The elapsed time between a cell's sending a request message and its acquisition of a channel.

therefore, the communication set up time is relatively small. The failure number of the algothm is kept at minimum. Moreover, the algorithm is simple to implement, and requires less storage and message overhead than existing algorithms.

II. Previous Work

Prakash, Shivaratri, and Singhal (PK) [8] proposed an algorithm based on *deferral* technique used in [9]. Choy and Singh (CS) [3] presented the algorithm that reduces the number of channel transfers in PK. Both PK and CS fall into the *search* category [4] since each MSS does not maintain the information about the channel being used by its neighbors. When a channel is needed (no available channels in the locally allocated set), the MSS searches all neighboring cells to compute the set of currently available channels.

Instead of gathering information each time a channel is needed, each MSS could maintain a set of available[2] channels by informing cells in its interference neighborhood each time it acquires and releases a channel. To request a channel, each cell must send a request message to each cell in its interference neighborhood. This is known as the *update* scheme. This scheme reduces acquisition delay at the expense of higher message complexity. Dong and Lai (DL) [4] proposed an update algorithm. Each cell sends request messages only to a small subset of its interference neighborhood, depending on the channel being requested.

Garg et al. [5] proposed two update algorithms. In the first algorithm ($G1$), the synchronization mechanism in [2] is employed. The second algorithm ($G2$) reduces the acquisition delay imposed in $G1$. However, no conflict resolution is used. Therefore a channel selected by two neighbors could potentially be dropped by both of them.

Our algorithm (QB) uses Maekawa's technique [7] to reduce message complexity in update algorithms. Like DL, request messages in QB are sent to only a small subset of cells in the interference neighborhood. However, this subset does not depend on the channels being requested. QB improves $G2$ by reducing the number of messages required per acquisition. We also use Lamport's timestamp [6] to totally order requests for channels among neighboring cells. Therefore, conflicts between any two cells within each other interference neighborhood are always resolved.

III. System Model

The allocated radio spectrum is divided into a number of channels. Each cell i is logically modelled as a hexagon of radius r with six neighbors Figure 1 displays a cellular network model.

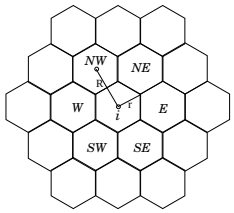

Figure 1: Hexagonal Cellular Model

The distance between cells is defined as the distance between the two centers. If R is the distance between two neighboring cells, then $R = \sqrt{3}r$. A channel can be reused in any two cells if the distance between them is at least D_{min}. We say that two cells are in each other *interference neighborhood* if the distance between them is less than D_{min}. Note that D_{min} must be greater than R.

We also assume that all MSSs are connected and every message sent between them is eventually received in FIFO order. Each MSS maintains Lamport's logical clock [6] so that events at different MSSs are totally ordered.

Given a cell c, $IN(c)$ denotes the set of all cells whose distance to c is less than D_{min}. Formally, $IN(c) = \{c' | dist(c, c') < D_{min}\}$, where $dist(c, c')$ denotes the distance between c and c'. We use n_{if} to denote $|IN(c)|$. We say that two cells, i, j, interfere with each other if the distance between them is less than D_{min}. We also say that i is in the interference neighborhood of j if $i \in IN(j)$. N is used to denote the total number of cells in the system.

The goal of any channel allocation algorithm is to ensure that no two interfering cells simultaneously use the same channel. To achieve this goal, any two requests from two interfering cells must be known to at least one of the arbitrators. If we assume that cell i obtains a permission from each member of a set S_i (*request set*) such that $S_i \subset IN(i)$, there must be at least one common cell between a pair of S_i and S_j for any two interfering cells i, j. We refer to this property as the pairwise nonnull intersection property ($\mathcal{PN}$). We use n_s to denote the size of the request set.

IV. The 3-Cell Cluster System

We here present a scheme to construct the request set such that $\mathcal{PN}$ is satisfied for the 3-cell cluster system. In this system, a channel cannot be reused within the same cell or the neighboring cells. This is because $R < D_{min} \leq \sqrt{3}R$. Consequently, there are exactly 6 members in each cell's interference neighborhood. To satisfy $\mathcal{PN}$, we assign to each cell i the

[2] Channels that cells can use without co-channel interference

following request set: $S_i = \{i, NW_i, SW_i, E_i\}$. We call this 1-*hop request set*. It is easy to see that the following property can be derived from 1-hop request set. Let S_{ij} denote $S_i \cap S_j$.

A. The Algorithm

Let $\mathcal{C}$ denote the set of channels shared by all cells in the system. Each cell c maintains two sets, U_c and I_c. U_c is the set of channels currently used by c. I_c is the set of interfered channels at c. Initially, every channel $k \in \mathcal{C}$ is available for use by c. A channel becomes an *interfered* channel for c if it is acquired by a cell in $IN(c)$.

When a cell needs a channel to support a communication session, it selects a channel k from $(\mathcal{C} - (I_c \cup U_c))$ based on the underlying channel allocation strategy. To acquire k, a cell sends a request message ($REQ(k)$) to each cell in its request set except itself. Once c receives a grant ($GRNT(k)$) from every cell in its request set, and if at this moment k does not belong to I_c, then channel k can be used to support a communication session in cell c, and is added into U_c. Next, c sends an acquisition message ($ACQ(k)$) to each cell in its interference neighborhood except those in its request set to inform about its use of channel k. When a cell receives $ACQ(k)$ message, k is added into its set I.

On receiving a request for channel k', c replies with a reject message (REJ) if either c is using this channel or c is also requesting k' with a smaller timestamp. Otherwise, c replies with a grant message ($GRNT$), and k' is added into I_c. Note that k' is added into I_c immediately after c has granted the request for k'. When the communication session using k is terminated, c sends a $REL\_TERM(k)$ message to each cell in I_c.

A cell realizes that its acquisition of channel k is failed when (1) a $REJ(k)$ has been received, or (2) a grant has been received from each cell in S_c but $k \in I_c$. Once the acquisition of channel k is failed, c sends release messages ($REL(k)$) to those cells that have already granted k to c. On receiving $REL(k)$ or $REL\_TERM(k)$, k is removed from I_c.

In [1], we prove that the proposed algorithm is save, that is, no two cells within each other's interference area concurrently use the same channel. We also prove that the algorithm is deadlock free.

B. Performance Analysis

If a cell has to make m attempts before it finally acquires a channel, the number of messages required per a channel acquisition is at most $2n_s + 3n_s(m-1) + (n_{if} - n_s) + n_{if}$.

Let D denote the average communication delay between MSSs. The average acquisition delay for the algorithm is $2mD$. The failure number of the 3-cell cluster algorithm is 3 since each support station receives request messages from exactly three neighboring cells.

V. Generalization

In this section, we present a general scheme to construct a request set for any given D_{min} and a general distributed channel allocation algorithm.

Let r_i^j denote the set of cells that are j hops away from i in the east, northwest, and southwest directions relative to i. The *n-hop* request set of cell i (denoted by S_i^n) is defined as follows:

$$S_i^n = i \cup r_i^1 \cup \ldots \cup r_i^n$$

Figure 2 shows S_{ij} for 2-hop request set. In the

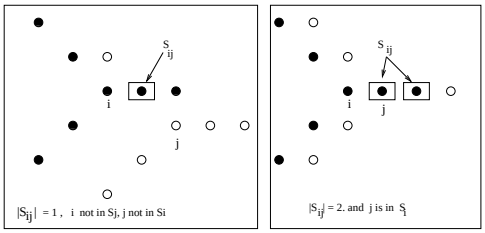

Figure 2: S_{ij} in 2-hop request set

previous section, we have shown that 1-hop request set satisfies $\mathcal{PN}$ when $R < D_{min} \leq \sqrt{3}R$. We show in [1] that the n-hop request set satisfies $\mathcal{PN}$ if $nR < D_{min} \leq (n+1)R$, for any integer $n \geq 1$.

A. The Generalized Algorithm

Unlike the previous algorithm, the response of each request can be either $GRNT$, C_GRNT, and REJ. A cell replies with C_GRNT to the request from cell i for a channel k if it has given at least one grant to another request from the cell within the interference neighborhood of i with a bigger timestamp. The ID's of those cells are also sent along with C_GRNT. A cell replies a channel k's request from cell i with $GRNT$ if either it is not using k, or it is not requesting k with a smaller timestamp, or it never grants the request of k to any cells that interfere with i before. Otherwise, the cell replies with REJ.

If i receives $C\_GRNT(W, k)$ from p, it is also required to check if $w \in S_i$ or $i \in S_w$, for each $w \in W$. This prevents co-channel interference because receiving $C\_GRNT(W, k)$ from p implies that the request from each cell $w \in W$ have been granted by p and the request timestamp of w is greater than that of i. If $w \in S_i$ or $i \in S_w$, then w will receive i's request or i will receive w's request, respectively. In either case, w's request will be failed given that i's request is not

rejected by w. Checking $i \in S_w$ can be done easily by calculating the distance and direction between i and w. Correctness proofs are presented in [1].

B. Performance Analysis

If a cell takes m requests before it acquires a channel, the average delay is $2Dm$. The number of messages becomes $2n_s + 3(m-1)n_s + 2n_{if}$. It is easy to see that the generalized algorithm has n_s failure number.

VI. Discussion

A. Comparison with Search Algorithms

Due to deferring mechanism used to resolve conflicts, acquisition delay in search algorithms significantly increases as traffic in the network grows. The size of the messages used in search algorithms is however larger than that of update algorithms. CS also suffers from the high acquisition delay under high traffic load due to the locking mechanism used to avoid co-channel interference and starvation.

B. Comparison with Update Algorithms

Under high traffic load, the response time of $G1$ is high since no two neighboring cells can choose frequencies at the same time. The acquisition delay of $G1$ is therefore significantly higher than that of QB in high traffic load. $G1$ requires $O(n_{if})$ messages per acquisition. However, $G1$ is starvation-free. $G2$ uses no conflict resolution mechanism, therefore, channels picked by two neighbors could potentially be dropped by both of them. The failure number of $G2$ is n_{if}. The number of messages required per acquistion is $O(n_{if})$

The detailed comparison between QB and DL is as follows. The memory overhead required in DL is higher than that required in QB. This is because the request set used in DL depends on the channel being requested. The size of the request set used in their algorithm is lower than ours. However, our request messages in the n-hop request set are only sent to cells at most n hops away in exactly three directions. The routing for our request messages is therefore simpler. The channel requests in DL can be considered failed due to message transmission delay. This is not the case in QB. DL is designed to work with a specific channel reuse pattern. QB can be used with any dynamic or hybrid allocation strategies. The failure number of DL is n_{if}.

Remark

Livelock: In the generalized algorithm, it is possible that $k \geq 3$ cells in each other's interference neighborhood request for the same available channel, but none of them will succeed. We call this *livelock* It can also occur in DL and PK.

Starvation: In theory, it could occur in every algorithm except CS and $G1$ that some cells may experience consecutive request failures or never be able to acquire a channel to support a call (even though the channel being requested is changed every time the cell makes another request attempt). The experimental result shows that the possibility for livelock and starvation to occur is extremely low.

Table below summarizes characteristic of each algorithm under two criterion: failure number and livelock.

	PK	CS	DL	$G1$	$G2$	QB
Failure Number	N	n_{if}	n_{if}	n_{if}	n_{if}	n_s
Livelock	Yes	No*	Yes	No*	Yes	Yes

* The dining philosophers algorithm [2] is used.

Enhancement: A common weakness of update algorithms is high message complexity due to *acquisition* and *release* messages sent to each cell in the interference neighborhood. We can reduce message complexity in the generalized algorithm by sending ACQ and REL_TERM messages only to S_c rather than $IN(c)$, and in the 3-cell cluster algorithm by omitting ACQ messages and sending REL_TERM messages only to S_c.

VII. Performance Evaluation

A. Simulation Environment

The simulation is performed on a 3-cell cluster system containing 7×7 hexagonal cells. The frequency band allocated to the system is divided into 400 independent channels. We assume that the one-way communication delay between any two MSSs is exponentially distributed with mean 100 milliseconds. The duration of a communication session during which a channel is in use is exponentially distributed with mean 120 seconds. The arrival of requests at each support station is modeled as a Poisson process with $1/\lambda$ calls per minute.

For each algorithm, two sets of experiments are performed. The first assumes a uniform arrival rate, that is, λ is constant over the entire region. The second assumes non-uniform traffic distribution, that is, the arrival process in each cell can be in either *normal* or *hot* state. The arrival rate in hot state is 5 times higher than normal state. The period of being in hot and normal state are exponentially distributed with mean 180 and 1800 seconds, respectively.

Handoff calls[3] and new calls are processed with the same priority. Each run executes 24,500 calls, but data was collected after 12,250 calls had been processed in order to eliminate the impact of startup transients. The mean value of five such runs, each

[3] the ongoing calls which are transferred from one cell to another due to the mobility of MHs

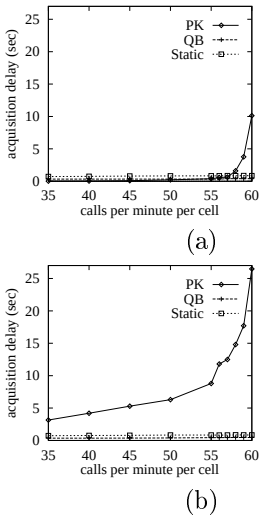

Figure 3: Average Acquisition Delay. (a) Uniform traffic, (b) Non-uniform traffic

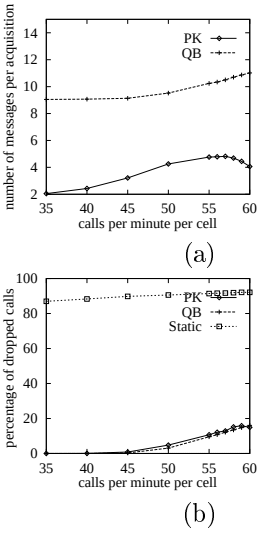

Figure 4: (a) Average Number of Messages used per Acquisition. (b)Percentage of Dropped Calls.

with a different random number seed, corresponds to a single data in the figures.

B. *Results*

We plot acquisition delay, the number of messages used per acquisition, and percentage of dropped calls from QB against those from PK, and static assignment algorithm(channels are preallocated uniformly). Figure 3 presents the average acquisition delay under uniform and non-uniform distribution. Due to space limitation, we only display here the average number of messages used per acquisition and percentage of dropped calls under uniform traffic distribution in Figure 4(a) and Figure 4(b), respectively.

VIII. Conclusion

We have presented an efficient distributed algorithm for dynamic channel allocation based on the update approach. Quorums are employed to reduce message complexity from the basic update algorithm. The proposed algorithm requires less storage overhead and is simpler to implement than the existing update algorithms. The member of our request set of each cell is fixed, not dependent upon the channel being requested. For $nR < D_{min} < (n+1)R$, only $3n$ request messages are sent to cells at most n hops away. Our algorithm is also flexible in that it can be used with any dynamic or hybrid channel allocation strategies.

Under high traffic load, the acquisition delay of the proposed algorithm is in the same neighborhood as that of static algorithm, and is significantly lower than that of search algorithms. The proposed algorithm also achieves the high degree of fault tolerance. In particular, it has the lowest *failure number* of all the existing algorithms.

References

[1] C. Skawratananond and Vijay K. Garg. A Quorum-based Distributed Channel Allocation Algorithm for Mobile Systems. available at http://maple.ece.utexa.edu/∼chakarat

[2] Manhoi Choy and Ambuj K. Singh. Efficient fault tolerant algorithms for distributed resource allocation. In *ACM Transactions on Programming Languages and Systems*, 17(4):535-559, 1995.

[3] Manhoi Choy and Ambuj K. Singh. Efficient Distributed Algorithms for Dynamic Channel Assignment. In *The Seventh IEEE International Symposium on Personal, Indoor and Mobile Rodio Communications*, pages 208-212, October 1996.

[4] Xuefeng Dong and Ten H. Lai. Distributed Dynamic Carrier Allocation in Mobile Cellular Networks: Search vs. Update. In *Proceedings of the 17th International Conference on Distributed Computing Systems*, pages 108-115, 1997.

[5] N. Garg, M. Papatriantafilou, and P. Tsigas. Distributed List Coloring: How to Dynamically Allocate Frequencies to Mobile Base Stations. In *Symposium on Parallel and Distributed Processing 1996*, pages 18-25.

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

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

[8] Ravi Prakash, N. G. 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.

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