TECHNION - Israel Institute of Technology
Computer Science Department

FAIR ALGORITHMS FOR MAXIMAL LINK ACTIVATION IN MULTI-HOP RADIO NETWORKS

by

I. Chlamtac and A. Lerner

Technical Report #397
March 1986
FAIR ALGORITHMS FOR MAXIMAL LINK ACTIVATION.
IN MULTI-HOP RADIO NETWORKS

by

Imrich Chlamtac and Anat Lerner

Computer Science Department
Technion --Israel Institute of Technology
Haifa 32000, Israel

ABSTRACT

We present a distributed algorithm for obtaining a fair timeslot allocation for link activation in a multi-hop radio network. We introduce the concept of maximal fairness in which the termination of a fair allocation algorithm is related to maximal reuse of the channel under a given fairness metric. The fairness metric can be freely interpreted as the expected link traffic load, demands, link priorities, etc. Since respective demands for timeslot allocation will not necessarily be equal, we define fairness in terms of the closeness of allocation to respective link demands while preserving the collision free property. The algorithm can be used in conjunction with existing link activation algorithms to provide a fairer and fuller utilization of the channel.
1. INTRODUCTION

One of many difficult problems associated with multi-hop radio networks is the finding of an efficient way of changing the unreliable multi-access channel into a reliable communication medium with efficient spatial reuse. A well documented and practical approach is based on a time oriented assignment of transmission rights. In the algorithms associated with this approach [2,3,8,11], transmissions executed according to the assignment, are guaranteed to be delivered reliably, i.e., without collision to their destinations. These algorithms, also termed link activation algorithms, attempt to assign timeslots for activation of links forming a short TDMA cycle such that spatial reuse of the radio channel is obtained. In each TDMA cycle, one transmission on each link is ensured [2,3]. As the problem of minimizing the length of the TDMA cycle in a multi-hop radio network was proven to be NP-Hard [1], the existing heuristic solutions may lead to unused timeslots. Furthermore, even if for a given network an optimal, i.e., shortest cycle allocation is computed, it may still be possible for timeslots to be further utilized by additional collision free activations of links in the same cycle.

In this paper we propose a solution for obtaining a link allocation in which all timeslots are fairly and maximally used. The proposed definition of "maximal fairness" differs from past fairness definitions [6,7] in accounting for maximality of channel reuse and in dealing with the integer allocation of timeslots. We propose centralized and distributed algorithms which construct such solution starting from any given TDMA cycle length. We show that the algorithms are fair with respect to allocating timeslots proportionally to the demand metric on each link. The algorithms are maximal in the sense that no further allocation is possible without violating fairness. The qualification of link demands, with respect to which the allocation is to be maximally fair, can be interpreted freely and can represent, for instance, expected traffic load on each link, link priorities, etc.
2. MODEL DESCRIPTION

For modeling a multi-hop radio network we use an undirected graph \( G(V,E) \), where a vertex \( v \in V \) represents a node in the network and an edge \( (v1,v2) \in E \) signifies that nodes \( v1 \) and \( v2 \) are in line of sight and within range of each other. We say that nodes \( v1 \) and \( v2 \) are neighbors or one hop away. As in previous models we shall assume that network nodes are synchronized and a message transmission can be initiated at the beginning of a constant duration timeslot \([2,3]\). The collection of all distinct timeslots defines a TDMA cycle with a well defined and commonly synchronized starting point.

For representation of directional transmission activity (source-destination) in the broadcast radio environment we shall say that an edge \( (v1,v2) \in E \) defines two links \( l_{v1,v2} : (v1 \rightarrow v2) \) and \( l_{v2,v1} : (v2 \rightarrow v1) \), with the set of all links denoted by \( E_{dir} \).

For rigorously defining a collision in a multihop radio network we shall say that links \( l_{v1,v2}, l_{v3,v4} \in E_{dir} \) are competing links if:

(a) \( v1 = v3 \) or \( v1 = v4 \) or \( v2 = v3 \) or \( v2 = v4 \), \{they have a common node\}

or

(b) \( (v1 \rightarrow v4) \in E_{dir} \) or \( (v2 \rightarrow v3) \in E_{dir} \), \{a transmitting node is a neighbor of a receiving node\}

We can now state that a collision in the radio network occurs when two or more competing links are activated in the same timeslot. In other words, consistently with prevailing radio network models, a collision in a given timeslot stands for the inability of a node to receive a message destined to it due to: 1) being a neighbor of more than one transmitting node, or 2) due to transmission of its own message.

With the given model we can now specify a timeslot as allocatable to link \( l_{ij} \) if there are no competing links already activated in that timeslot.
For convenience of subsequent definitions and analysis we further denote \( \tau \) to be the maximum number of neighbors of any network node. We denote by \( L \) the length of the TDMA cycle and by \( \#l_{ij} \) the number of timeslots assigned for transmission on link \( l_{ij} \) in a single TDMA cycle. Lastly we denote by \( \lambda_{ij} \) the positive value assigned to link \( l_{ij} \) according to the chosen allocation metric representing the demand associated with that link.

**Definition 2.1:** A single-slot-per-link allocation is the assignment of exactly one timeslot to each link in a TDMA cycle. Notice that the allocation can be represented as a function from \( E^\text{dir} \) to an integer number (timeslot) between 1 and \( L \).

**Definition 2.2:** Slot allocation is the assignment of at least one timeslot to each link in a TDMA cycle. We can represent such allocation as a function from \( E^\text{dir} \) to the power set of integer numbers between 1 and \( L \) (\( 2^{[1..L]} \)).

**Definition 2.3:** Slot allocation \( B \) is collision free if for every timeslot \( s \), there is no activation of competing links. Using the functional representation of slot allocation, \( B \) is collision free if for every competing pair of links, \( l_{ij}, l_{km}, B(l_{ij}) \cap B(l_{km}) = \emptyset \).

### 3. FAIRNESS AND MAXIMAL FAIRNESS

Applying existing definitions of fairness metrics [6,7] to allocation of a shared resource, we obtain the following criterion for determining whether an allocation obtained from a given collision free slot allocation \( B \) is fair.

A slot allocation \( B \) is fair if for a given channel (resource) demand matrix \( A, A[i,j] = \lambda_{ij} \) and for every two links (users) \( l_{ij}, l_{km} \in E^\text{dir} \), the respective fractions of allocation to demand are equal [8]:
Notice however that, since the channel resource is allocated in integer number of slots, then according to this criterion a fair allocation may not exist or its preservation may severely restrict the allocation of unused TDMA timeslots. For example, a fair distribution of two slots among two links \( l_{ij}, l_{ji} \), such that \( \lambda_{ij} = 1 \) and \( \lambda_{ji} = 4 \) is not feasible.

To solve this problem we define a fairness metric such that instead of comparing the allocation to demand ratio of each pair \( \frac{\#l_{ij}}{\lambda_{ij}}, \frac{\#l_{km}}{\lambda_{km}} \) according to the above definition of fairness, we consider the combined resources allocated to both links \( \#l_{ij} + \#l_{km} \). We say that the allocation is fair if this sum is distributed proportionally to the respective demands of each link under the constraint of integer slot allocation.

We use the following \textit{round} operation:

\[
\text{round} \cdot x \cdot a \begin{cases} 
  x & a \leq 1/2 \\
  x+1 & a > 1/2
\end{cases}
\]

Notice that for \( a=1/2 \) both values are acceptable to obtain the following fairness criterion.

**Definition 3.1:** Slots allocation is fair iff. \( \forall l_{ij}, l_{km} \in \mathbb{P}_{\text{fair}} \):

\[
\#l_{ij} = \begin{cases} 
  \text{round} \left( \frac{\lambda_{ij}(\#l_{ij} + \#l_{km})}{\lambda_{ij} + \lambda_{km}} \right) & \#l_{ij} + \#l_{km} > x \geq 1 \\
  1 & x = 0 \\
  \#l_{ij} + \#l_{km} - 1 & x = \#l_{ij} + \#l_{km}
\end{cases}
\]

The above definition of fairness allows for integer resource allocation. However similarly to prevailing definitions [6,7] it still does not consider the extent to which the total amount of resource has been shared. In this way every single-
slot-per-link allocation will be 100% fair while in general network topologies, such allocation may utilize only a small part of the channel resource.

Due to the importance of maximizing the utilization of radio channel, we therefore wish to extend our definition of fairness to encompass the level of channel utilization. We wish to obtain a maximality property, i.e., an allocation in which no further slot assignment can be made without violating the fairness criterion given in Definition 3.1. Therefore we say:

Definition 3.2: A slot allocation $C$ is maximally fair if:

a) $C$ is fair by Definition 3.1, and

b) for every slot allocation $C'$ that contains at least the same allocation given by $C$ (i.e., $C(l_j) \subseteq C'(l_j)$, $\forall l_j \in L^{dr}$), $C' = C$ or $C'$ is not fair.

In other words, any additional slot assignment violates fairness.

Notice that Definitions 3.1 and 3.2 imply that the fairness property must be preserved globally in the network, i.e., it holds for every, not necessarily adjacent, pair of links. This global attribute is important to prevent the following chain-unfairness phenomenon. Consider a network where $\lambda_{ij}$'s are equal for all links. Then, the allocation of timeslots shown in figure 1 is fair in the local sense, i.e., it holds for all outgoing and incoming links of each node. While algorithms which preserve only local fairness are clearly easier to construct (only allocations made to links of neighboring nodes must be considered) the chain unfairness example clearly demonstrates the inadequacy of such approach.

4. MAXIMALLY FAIR ALGORITHM

When requiring the preservation of the fairness property globally for every pair of links in the network, the main problem in deriving an efficient centralized or distributed algorithm directly from Definition 3.2, comes from the need to
perform comparisons between all pairs of links \((E, E^2)\) in the network prior to each timeslot allocation.

We therefore next define a measure which can be evaluated independently for each link using only local information but whose use preserves the fairness property globally.

Definition 4.1: We define a saturation measure \(s(l_{ij})\), as follows:

\[
s(l_{ij}) = \frac{n_{ij}}{\lambda_{ij}} + \frac{1}{2\lambda_{ij}}.
\]

Intuitively this measure is calculated as the average between the current allocation to demand ratio of the link \(\frac{n_{ij}}{\lambda_{ij}}\), and the ratio resulting from the allocation of an additional slot to link \(l_{ij}\), \(\frac{n_{ij}+1}{\lambda_{ij}}\).

Using the link saturation measure definition, we can formally describe the Centralized Maximally Fair (CMF) algorithm:

\[
\begin{aligned}
&\text{begin} \\
&\text{terminate:=false} \{\text{termination indicator}\} \\
&\text{while not terminate do} \\
&\quad\text{smallest measure:= min}_{l_{ij} \in E} s(l_{ij}) \\
&\quad\text{For all links } l_{km} \text{ s.t. } s(l_{km}) = \text{smallest measure} \text{ do} \\
&\quad\quad\text{allocate a timeslot \{consistent with Def. 2.3\}} \\
&\quad\quad\text{If no slot is allocatable then} \\
&\quad\quad\quad\text{terminate:=true} \\
&\quad\quad\text{else } s(l_{km}):=s(l_{km}) + \frac{1}{\lambda_{km}} \{\text{calculating the new saturation measure of link } l_{km}\} \\
&\text{end} \\
&\text{end} \\
&\text{end;}
\end{aligned}
\]

Comment: We assume that on initialization of the CMF algorithm a single-slot-per-link allocation e.g. such as [2] is given.
Theorem 4.1: The CMF algorithm is (globally) fair.

Proof of Theorem 4.1: We prove by induction on the execution steps, where each step includes one allocation. Initially, \( \#l_{ij} = 1, \forall l_{ij} \in E_{dir} \) which is a fair distribution. Assuming that by the end of the \( p \)-th step, \( \forall l_{ij}, l_{km} \in E_{dir} \)

\[
\#l_{ij} + \frac{1}{\lambda_{ij}} \leq \frac{\#l_{km} + 1}{\lambda_{km}} + \frac{1}{2\lambda_{km}}.
\]

We have to prove that:

\[
x = \text{round}\left( \frac{\lambda_{ij} (\#l_{ij} + \#l_{km} + 1)}{\lambda_{ij} + \lambda_{km}} \right)
\]

\[
\#l_{ij} + 1 = \begin{cases} 
1 & \text{if } x = 0 \\
\#l_{ij} + \#l_{km} - 1 & \text{if } x = \#l_{ij} + \#l_{km}
\end{cases}
\]

(The post allocation state remains fair.)

a) If \( \#l_{ij} = \text{round}\left( \frac{\lambda_{ij} (\#l_{ij} + \#l_{km})}{\lambda_{ij} + \lambda_{km}} \right) \) then

(i) \( \#l_{ij} + 1 \geq \text{round}\left( \frac{\lambda_{ij} (\#l_{ij} + \#l_{km} + 1)}{\lambda_{ij} + \lambda_{km}} \right) \) (trivially) and

(ii) \( \frac{\#l_{ij}}{\lambda_{ij}} + \frac{1}{2\lambda_{ij}} \leq \frac{\#l_{km}}{\lambda_{km}} + \frac{1}{2\lambda_{km}} \)

\[
\Rightarrow \#l_{km}\lambda_{ij} \geq \#l_{ij}\lambda_{km} + \frac{\lambda_{km} - \lambda_{ij}}{2}
\]

\[
\Rightarrow \frac{\lambda_{ij} (\#l_{ij} + \#l_{km})}{\lambda_{ij} + \lambda_{km}} \geq \#l_{ij} + \frac{\lambda_{km} - \lambda_{ij}}{2(\lambda_{km} + \lambda_{ij})}
\]

\[
\Rightarrow \frac{\lambda_{ij} (\#l_{ij} + \#l_{km} + 1)}{\lambda_{ij} + \lambda_{km}} \geq \#l_{ij} + \frac{1}{2}
\]

If \( s(l_{ij}) < s(l_{km}) \) then \( \text{round}\left( \frac{\lambda_{ij} (\#l_{ij} + \#l_{km} + 1)}{\lambda_{ij} + \lambda_{km}} \right) \leq \#l_{ij} + 1 \).
If $s(l_y) = s(l_{km})$ then round $\left\lfloor \frac{\lambda_y (\#l_{ij} + \#l_{km} + 1)}{\lambda_y + \lambda_{km}} \right\rfloor = \left\lfloor \#l_{ij} \right\rfloor$.  

Consequently, an additional timeslot allocated to each one of the links with the same smallest saturation measure preserves fairness.

b) If $\#l_{ij} = 1$ and for some $l_{km}$ round $\left\lfloor \frac{\lambda_y (\#l_{km} + 1)}{\lambda_y + \lambda_{km}} \right\rfloor = 0$ then,  

\[
\frac{1}{2} > \frac{\lambda_y (\#l_{km} + 1)}{\lambda_y + \lambda_{km}} \\
=> \lambda_y (2\#l_{km} + 1) < \lambda_{km} \\
=> s(l_y) = \frac{3}{2\lambda_y} \geq 3 \frac{2\#l_{km} + 1}{\lambda_{km}} = 3s(l_{km}) \\
=> s(l_y) > s(l_{km})
\]

and thus $l_y$ will not be a candidate for allocation.

c) If $\#l_{km} = 1$ and round $\left\lfloor \frac{\lambda_{km} (\#l_{ij} + \#l_{km})}{\lambda_{km} + \lambda_{ij}} \right\rfloor = 0$, then by allocating an additional timeslot for link $l_y$, round $\left\lfloor \frac{\lambda_{km} (\#l_{ij} + \#l_{km} + 1)}{\lambda_{km} + \lambda_{ij}} \right\rfloor = 1$ and thus fairness is maintained. []

Theorem 4.2: The CMF algorithm is maximally fair.

Proof of Theorem 4.2: From Theorem 4.1, the CMF algorithm is fair. To prove maximality we have to show that any additional slot allocation violates fairness.

Let $l_y$ be a link with the smallest saturation measure, s.t. there is no allocatable slot for link $l_y$. Let $l_{km}$ be a link, s.t. $s(l_{km}) > s(l_y)$. By allocating an additional slot for link $l_{km}$:

a) If $\#l_{ij} = \text{round} \left\lfloor \frac{\lambda_y (\#l_{ij} + \#l_{km})}{\lambda_y + \lambda_{km}} \right\rfloor$ then from Theorem 4.1, $\#l_{ij} + 1 = \text{round} \left\lfloor \frac{\lambda_y (\#l_{ij} + \#l_{km} + 1)}{\lambda_y + \lambda_{km}} \right\rfloor$ and thus $\#l_y < \text{round} \left\lfloor \frac{\lambda_y (\#l_{ij} + \#l_{km} + 1)}{\lambda_{km} + \lambda_{ij}} \right\rfloor$. 

Technion - Computer Science Department - Technical Report CS0397 - 1986
b) If \( \#l_{km} = 1 \) and \( \text{round}\left(\frac{\lambda_{km}(\#l_{ij} + \#l_{km})}{\lambda_{km} + \lambda_{ij}}\right) = 0 \) then from Theorem 4.1
\[\text{round}\left(\frac{\lambda_{km}(\#l_{ij} + \#l_{km} + 1)}{\lambda_{km} + \lambda_{ij}}\right) \leq 1.\]

In both cases by allocating an additional slot to \( l_{km} \), fairness is no longer preserved. []

**Theorem 4.3:** The computation complexity of the CMF algorithm is \( O(|V| L \cdot \log |E| + L) \):

**Proof of Theorem 4.3:** The main allocation loop (while not terminate) can be repeated no more than \(|V| L\) times.

Each time at least one link is being allocated a slot. No more than \( \frac{|V|}{2} \) links can be activated at the same time and the total amount of timeslots is \( L \). For a given sorted list, the finding of minimum takes \( O(1) \) time. The allocation process takes \( O(L) \) and finding the proper place in the sorted list of the link with the new saturation measure takes \( O(\log E) \) steps. The complexity of \( O(|V| L \cdot [L + \log E]) \) is thus obtained. []

5. THE DISTRIBUTED MAXIMALLY FAIR ALGORITHM (DMF)

We present the DMF algorithm which is distributed in the sense that it requires no centrally stored information, no global knowledge of the topology and there is no special central station. The algorithm is continuously adaptive to topological changes, can be executed in parallel and its execution does not disrupt data transmission of other than the position changing node.

For simplicity of presentation, we first describe the DMF operation in a static network. The handling of topological changes will be described in Section 5.3.
The principle of the distributed algorithm is to preserve the allocation order of the centralized approach. This is done by locating each time a node that has outgoing links with the smallest saturation measure and finding allocatable slots to those links. We therefore define node saturation measure as follows:

**Definition 5.1.** Node $i$ saturation measure is defined as $S(i) = \min_{l_j \in E_{i}} s(l_j)$, i.e., the minimum of saturation measures of all $i$'s outgoing links.

The distributed approach consists of two phases. The first phase constructs an augmented spanning tree in which each node knows its father, its sons and $S^f(i)$ the smallest saturation measure of all nodes in the subtree rooted at each son $i$.

The second phase uses the information obtained from the first phase of the algorithm, to pass an allocation rights token each time to all nodes with the current minimum saturation measure in turn. When such a node, say $i$, receives the allocation rights token, it allocates timeslots to all its outgoing links $l_j$ for which $s(l_j) = S(i)$ and updates its $S(i)$ and $S^f(i)$ measures.

The algorithm terminates when no further allocatable slots can be found for links with current minimum $s(l_j)$ value.

**5.1 DMF Algorithm Description**

*Phase one: Tree Construction*

Since the process of locating the node with the minimum $S$ measure is repeated, we seek an efficient construction for locating the minimum. We thus construct a spanning tree using a distributed algorithm which hierarchically stores the local minimum for all sons in each subtree, $S^f$. In this way, using the local information, the minimum $S(j)$ node can be located in $O(\text{height})$ of the tree.
To minimize the tree height we seek an algorithm that constructs shortest path from a given root to all other nodes. The algorithm must be adaptable to topological changes. Among the many existing tree constructing algorithms [5,12,13,19], we have chosen the Propagation of Information with Feedback [PIF] algorithm [12]. The PIF algorithm constructs a shortest spanning tree from a given root thus minimizing the time needed for locating \( \min S(j) \) and can be extended to networks with changing topology. Secondly, we can use an existing single-slot-per-link allocation e.g. [2], to pass control messages without collisions thus yielding fast execution of tree construction. We therefore augment the PIF algorithm to serve as the basis for the first phase of the DMF algorithm as follows:

The algorithm constructs the spanning tree in two steps termed uptree (from root to leaves) and downtree. The uptree step is used for determination of each node's father and is initiated by the root node by sending an uptree message to all its neighbors. When node \( j \) receives the first such message from a neighboring node \( i \), it records \( i \) as its father in the tree and forwards the message to all its neighbors except node \( i \). The downtree step is initiated by the leaves who send their \( S \) measure to their father on reception of uptree messages from all their neighbors. When a node, say \( j \), receives downtree messages from all its sons (and uptree messages from all remaining neighboring nodes) it calculates its \( S^t(j) \) value computed as \( \min \{S(j), \min_{k \in \text{son}(j)} S^t(k)\} \). The calculated \( S^t(j) \) value is then sent downtree to its father (node \( i \)). Phase one of the DMF is terminated when the root receives \( S^t(\cdot) \) measures from all its sons. At this point the following information is collected at each node \( j \): 1) \( j \) 's father in the spanning tree, 2) list of \( j \) 's sons with their respective \( S_t \) measures, and 3) \( j \) 's \( S^t(j) \) and \( S(j) \) measures.
Phase two: Maximal Fair Slot Allocation

The following description is consistent with the formal DMF algorithm specification given in the Appendix.

The information serves as the input to the second phase of the DMF at node $i$ is:

1) A single-slot-per-link allocation providing:
   a) LOCAL TDMA: transmission slots used by node $i$'s outgoing and incoming links, and slots used by competing links ("restrictions").
   b) NEIGHBORS _STATES: the LOCAL TDMA of each neighboring node.
   c) The TDMA cycle length - L.
2) The $i$-th row of the demand matrix $A (\lambda_{ij}$ for all outgoing links).
3) A list of $i$'s NEIGHBORS.
4) FATHER, $S^t(i)$ and SONS - with the $S^t$ measure of each son obtained from phase one.

Initialization: For every neighboring node $j$.
1. $s(l_{ij}) = \frac{3}{2\lambda_{ij}}$, obtained from assignment of $\#l_{ij} = 1$ in Definition 4.1.
2. $S(i)^t = \min_{l_{ij} \in S^t} s(l_{ij})$.
3. Termination flag = false, used for implementing distributed termination.

We assume that the root ($R$) is initially in possession of the allocation rights token. The root performs the same process as any node $i$ receiving the token. Since following phase, "one" each node is aware of the minimum saturation measure of each subtree rooted at its sons $S^t(j)$, the token can always be routed in the direction of a globally smallest saturation measure.

When a node receives the token in a PASS TOKEN message, it starts an allocation process by testing whether it itself qualifies for slot allocation, i.e., its own saturation measure is the minimal in its subtree $(S(\cdot) = S^t(\cdot))$ or whether it
should forward the token to one of its sons. Notice that multiple tokens can be
forwarded simultaneously to all sons having the minimum $S^t(\cdot)$ measure. For simplicity of presentation we restrict the following discussion to sequential token passing. Assume the token has reached a node, say $\mathbf{i}$, with a minimum saturation measure. To find an allocatable slot, node $\mathbf{i}$ first updates its NEIGHBORS_STATES information by issuing an UPDATE_REQUEST message following which it receives the LOCAL_TDMA_INFO message from all its neighbors. If, based on this updated information, an allocatable slot can be found for one or more outgoing links whose saturation measure $s(L_y)$ is minimal, node $\mathbf{i}$ updates $\#L_y$, $s(L_y)$ and $S(\mathbf{i})$ using Definitions 4.1 and 5.1 respectively and informs its neighbors of its new LOCAL_TDMA in a NEW_TDMA message. (Notice that the use of a separate NEW_TDMA message, in addition to the LOCAL_TDMA_INFO message described above, provides each node with a two hop TDMA information which is necessary and sufficient to ensure a collision free slot allocation).

If an allocatable slot is not found, a termination flag is raised. Node $\mathbf{i}$ next passes the token in a PASS_TOKEN message to a son $\mathbf{j}$ having the same $S^t(\mathbf{j})$ measure who repeats the same process and eventually returns the token to node $\mathbf{i}$ in a RETURN_TOKEN message. This procedure is repeated for each son with the same $S^t$ measure. When eventually, node $\mathbf{i}$ receives the RETURN_TOKEN message from the last such son (or when no PASS_TOKEN message was sent by node $\mathbf{i}$ at all), node $\mathbf{i}$ will update $S^t(\mathbf{i})$ (see phase one), and send a RETURN_TOKEN ($\mathbf{i}$, father($\mathbf{i}$), $S^t(\mathbf{i})$, termination flag) message to its father thus ending the allocation process.

For algorithm termination purposes it is important to notice that at this point the allocation process has been executed at all nodes, in $\mathbf{i}$'s subtree, having the same minimum saturation measure. Furthermore notice that as proven earlier the inability to allocate a slot to a link with a minimum $s(\cdot)$ measure implies that no further fair allocations to links with higher saturation measures are
possible. In the DMF algorithm, we take advantage of these two properties to obtain a distributed termination: let the termination flag of node \( i \) be assigned the OR value of all termination flags received by this node from its sons. If this value is true, the algorithm can be terminated at all nodes in node \( i \)'s subtree (using a TERMINATE message). The passing of this flag value by node \( i \) to its father enables further the termination to be gradually distributed in the network.

5.2 Demonstration

We consider the network of 5 nodes as shown in figure 2. The numbers on the arrows represent the demand matrix \( \Lambda \) of the respective links. The algorithm receives as input the single_slot_per_link allocation shown in table 1 with a TDMA cycle length of 10 timeslots.

For ease of demonstration we divide the execution of the algorithm into steps. Each step consists of the allocation of timeslots to all links with the same saturation measure which is the current minimum.

Initial values are as follows:

- \( s(l_{31}) = s(l_{13}) = 0.1875 \)
- \( s(l_{25}) = 0.3 \)
- \( s(l_{24}) = s(l_{52}) = s(l_{45}) = 0.5 \)
- \( s(l_{21}) = s(l_{35}) = s(l_{54}) = 0.75 \)
- \( s(l_{12}) = s(l_{53}) = s(l_{42}) = 1.5 \)

Step 1: Slots are allocated to links \( l_{13} \) and \( l_{31} \) (in that order). By the end of step 1, \( \#l_{31} = \#l_{13} = 2 \) and \( s(l_{31}) = s(l_{13}) = 0.3125 \).

Step 2: Link \( l_{25} \) has the current smallest saturation measure. After step 2, \( \#l_{25} \) is changed to 2 and \( s(l_{25}) = 0.5 \).

Step 3: Allocation is made to links \( l_{13} \) and \( l_{31} \) respectively. After timeslot allocation the new value of \( \#l_{13} \) and \( \#l_{31} \) is 3 and \( s(l_{13}) = s(l_{31}) = 0.4375 \).

Step 4: Slots are allocated to links \( l_{13} \) and \( l_{31} \). After that step, \( \#l_{31} = \#l_{13} = 4 \) and
\[ s(I_{13}) = s(I_{21}) = 0.5625. \]

**Step 5:** Links \( I_{24}, I_{25}, I_{52} \) and \( I_{45} \) receive allocation rights in that order (determined by the spanning tree shown in figure 2). A slot is successfully allocated to link \( I_{52} \) while test for allocation on links \( I_{25}, I_{24} \) and \( I_{45} \) fails - no slot is allocatable for these links. Step 5 is the final step. The resulting allocation is shown in table 2.

### 5.3 Handling of Topological Changes

The objective of the DMF in a mobile environment is the continuous updating of the slot allocation which preserves the maximal fair allocation dynamically. The principle of the extended DMF is the operation in cycles triggered by topological changes. In each cycle, the global minimum saturation measure \( S_{\text{min}} \) is recomputed at all nodes. Given the new \( S_{\text{min}} \) each node deletes timeslots previously assigned to its outgoing links whenever the assignment violates the fairness with respect to the new \( S_{\text{min}} \) value. Starting from the obtained fair allocation, the DMF executes the second phase for maximally fair allocation as before. It is important to notice that with DMF execution being triggered by topological changes, fairness and maximality violations are not accumulated as in the case of algorithms with periodic slot reorganization. The proposed approach for dynamic reallocation thus provides continuous fair and maximal reuse of the channel without limiting the number of topological changes that can be handled in parallel through overlapping DMF executions and allowing the use of multiple tokens as suggested in Section 5.1. In the extended DMF we augment phase one of the algorithm to perform the following:

- **a)** continuous updating of the spanning tree structure,
- **b)** calculating the updated \( S^f \) measures at each node and
- **c)** providing a new fair allocation at the end of phase one as an input to the second phase of the DMF.
For task a) we employ the extension of the PIF algorithm for dynamic networks. For the sake of clarity, we describe its principles only as pertinent to the DMF execution. A complete description of the extended PIF algorithm is given in [12,13]. The central idea of the extended PIF algorithm is to trigger a new tree construction cycle whenever a topological change is detected for creating a spanning tree similarly to Section 5.1. A request for a new cycle can be sent by any node to the root. Cycles are numbered with increasing numbers and every node remembers the highest cycle number known to it so far. Since any topological change induces generation of a reorganization request, it is guaranteed that at least one of the messages referring to the same cycle number will indeed arrive at the root and a spanning tree construction will take place.

Task b) uses the down-tree step of the extended PIF for $S^t$ calculation similarly to the static case. If a topological change occurs during the execution of task b) at a given node, task execution is aborted. Notice that in such case a new task b) is guaranteed to be initiated by the leaves on termination of task a).

For task c) we apply the slot deletion process as follows. The root node detects unfairness in allocation when the new $S_{\text{min}}$ measure is smaller than the $S_{\text{min}}$ measure of the previous cycle. In that case the root initiates a correction process by broadcasting the new $S_{\text{min}}$ measure along the tree. Upon reception of $S_{\text{min}}$, each node deletes timeslots allocated to its outgoing links until for each link $L_j$, the deletion of an additional timeslot, would result in $S^t(L_j)$ measure smaller or equal to $S_{\text{min}}$. When a node, say $i$, learns of the completion of the deletion process in its subtree, it updates its $S^t(i)$ measure and sends it to its father on the tree. The deletion process is terminated when the root receives $S^t$ measures from all its sons, at which point a fair allocation is restored. In its second phase, the DMF algorithm builds on this fair allocation to perform a maximally fair allocation similarly to the static case.
6. PROPERTIES OF THE DMF ALGORITHM

We prove that the DMF algorithm is maximally fair while preserving the collision free property and derive the message complexity of the algorithm.

**Theorem 6.1:** The allocation made by the DMF algorithm is collision free.

*Proof of Theorem 6.1:* Prior to the allocation process at node $i$, node $i$ is aware of all transmission slots used by nodes up to two hops away (potentially competing links). Each node learns about transmission slots of neighboring nodes from an OWNVERSION message followed by an allocation. Therefore, at every point in time, each node has a complete version of slots used by neighboring nodes as senders. The complementary information about slots used by neighboring nodes as receivers is accomplished by the exchange of UPDATE REQUEST and LOCAL TDMA messages. Hence, prior to the actual allocation, each node is aware of all slots used by competing links.

**Theorem 6.2:** The DMF algorithm is fair.

*Proof of Theorem 6.2:* From Theorem 4.1 the strategy of allocating a slot to the link with the smallest saturation measure ensures a fair allocation. As each node $i$ is aware of $s(i_y)$ and of $S_i(k)$ for all $k \in SONS(i)$, the allocation rights token is passed to the first node in the path defined by the spanning tree with the minimum saturation measure $S$.

**Theorem 6.3:** The DMF algorithm is maximally fair.

*Proof of Theorem 6.3:* From Theorem 4.2, the termination on the condition that none of the link having the current minimum saturation measures in the network can be allocated an additional timeslot, ensures maximality.
The distributed termination at node \( i \) (i.e., till the next topological change) occurs: a) when a node possessing the allocation rights token and all the nodes in its subtree cannot allocate any timeslot preserving fairness. This is the only case where a TERMINATE message is initiated; or b) on reception of a TERMINATE message from the father. In both cases, when node \( i \) terminates, \( S(i) \) is larger than the current \( S_{\text{min}} \), or ii) \( S(i) = \text{current} \) \( S_{\text{min}} \) but no slot is allocatable for its outgoing links \( L'_i \), s.t. \( s(L'_i) = S(i) \). Thus maximality is obtained.

**Theorem 6.4:** The complexity of the number of messages passed by the DMF algorithm is \( O(|E| L H) \), with \( H \) the height of the spanning tree and \( L \) the length of the TDMA cycle.

**Proof of Theorem 6.4:** The spanning tree construction (phase one) takes \( O(|E|) \) messages. The process of searching for the node with the minimum measure is repeated no more than \( L \sqrt{V}/2 \) times. Since each search is bounded by the height of the tree \( (H) \), the complexity of \( O(|V| L \cdot H) \) is obtained. 

7. CONCLUSION

We have presented a definition of maximal fairness for allocation of timeslots to links in multi-hop radio networks proportionately to their demands. Centralized and distributed maximally fair algorithms were given which, using a local 'saturation' measure, provide a globally fair and maximal slot allocation. It was shown that the use of this measure greatly reduces the complexity of both algorithms. The presented algorithms assume only the existence of a single slot per link allocation at initialization time. Such initial allocation can be obtained from a number of existing link activation algorithms. Therefore, the proposed algorithms can be used for improving an existing allocation in terms of radio channel utilization and fairness from global network considerations.
ACKNOWLEDGMENT

We wish to thank Reuven Bar-Yehuda for his participation in constructive and helpful discussions.

REFERENCES


APPENDIX: Pseudo Code of the DMF Algorithm

For any node i:

VAR
NEIGHBOR_STATES, LOCAL_TDMA: array [1..L] of records: state:
   state description: [in LOCAL_TDMA node i keeps transmission and reception slots. In NEIGHBORS STATES, transmission and reception slots of neighboring nodes are kept].
LINKS: sorted linked list of records:
   id {id=1 represents link l_i}
   s {s(l_i)}
   λ {λ_i}
   #l {#l_i}
   the list is sorted by s
SONS: sorted linked list of records:
   id {son's identification number}
   S^i {S^i(id)}
FATHER: {node i's father}
S^i {S^i(i)}
Termination_flag: boolean
NEIGHBORS: {list of neighbors}

MESSAGES TYPE:
PASS_TOKEN(id1,id2) {the token is passed from node id1 to node id2}
UPDATE_REQUEST(id) {an UPDATE_REQUEST message}
LOCAL_TDMA_INFO(id,local_TDMA(id)) {reception of a LOCAL_TDMA_INFO message}
NEW_TDMA(id,local_TDMA(id)) {a NEW_TDMA message}
RETURN_TOKEN(id1,id2,S^i,termination_flag) {the token is returned from node id1 to its father node id2}
TERMINATE(id) {a broadcast message indicating termination}

Procedure ALLOCATE (l_i,term)
   {receives a link as an input and attempts to find an allocatable slot. Term is updated accordingly}
begin
   if there is an allocatable slot for l_i then begin
      update LOCAL_TDMA, NEIGHBORS_STATES;
      #l_i:=#l_i+1; s(l_i):=s(l_i)+1/λ_i;
      restore link l_i to LINKS using the new s(l_i) key; term:=false;
      S(i):=first(LINKS).s
   end else begin
      term:=true;
      pop(LINKS) {no further allocation attempts for link l_i will be made}
   end
end;
Procedure \texttt{TOKEN\_RECEIVED}

\{to be called on reception of the allocation rights token\}
begin
  \textbf{if} \(S(i) = S'(i)\) \textbf{then}
  begin
    \texttt{node }\texttt{i}\texttt{ qualifies for slot allocation}
    \texttt{done:=false} \{\texttt{no allocation has actually taken place yet}\}
    \texttt{send UPDATE\_REQUEST}(i)
    \texttt{for every }\texttt{w }\in \texttt{NEIGHBORS} \texttt{do}
    begin
      \texttt{wait for LOCAL\_TDMA\_INFO}(w, LOCAL\_TDMA(w))
      \texttt{update LOCAL\_TDMA, NEIGHBORS\_STATES}
    end
  \end
  \texttt{while first (LINKS).}S=S'(i) \texttt{do} \{\texttt{to be repeated for all links with the same saturation measure}\}
  begin
    \texttt{ALLOCATE}(k, \texttt{first(LINKS).}id, \texttt{term})
    \texttt{termination\_flag:=termination\_flag.OR.}\texttt{term}
    \texttt{If} \texttt{not} \texttt{term} \texttt{then done:=true}
  end
  \texttt{If} \texttt{done} \texttt{then} \{\texttt{at least one successful allocation}\}
  \texttt{send NEW\_TDMA (i,LOCAL\_TDMA)}
end; \{of allocation of slots for node \texttt{i}'s outgoing links\}

\texttt{while first (SONS).}S=S'(i) \texttt{do} \{\texttt{to be repeated for all links with the same saturation measure}\}
begin
  \texttt{pass token to sons with the same} \texttt{S'} \texttt{minimal measure}
  \texttt{send PASS\_TOKEN}(i,\texttt{first(SONS).}id)
  \texttt{wait for RETURN\_TOKEN}(\texttt{first(SONS).}id,i,S,term)
  \texttt{first(SONS).}S=S'
  \texttt{termination\_flag:=termination\_flag.OR.}\texttt{term}
  \texttt{restore first(SONS) to SONS using the new} \texttt{S'} \texttt{key}
end; \{of the use of token\}

\texttt{ST}(i):=\min(\texttt{first(SONS).}S', \texttt{S}(i))
\texttt{If FATHER }\neq \texttt{null} \texttt{then send RETURN\_TOKEN}(i,\texttt{FATHER}, \texttt{S'}(i), \texttt{termination\_flag})
\texttt{If} \texttt{termination\_flag} \texttt{and nonempty(SONS)}\texttt{then send TERMINATE}(i)
end; \{of TOKEN\_RECEIVED\}

\texttt{ON UPDATE\_REQUEST(id) do.}

\{node id requests node's} \texttt{i LOCAL\_TDMA\}
begin
  \texttt{send LOCAL\_TDMA\_INFO}(i,LOCAL\_TDMA)
end;

\texttt{ON NEW\_TDMA(id, id LOCAL\_TDMA) do}

\{reception of node's id new LOCAL\_TDMA\}
begin
  \texttt{update LOCAL\_TDMA, NEIGHBORS\_STATES}
end;
ON TERMINATE(j) do

begin
  If j=FATHER then
  begin
    stop-run:=true {termination of a father implies the
    termination of all nodes in its subtree}
    If nonempty(SONS) then
    send TERMINATE(i);
  end
end;

ON PASS_TOKEN(id1,id2) do

{reception of a PASS_TOKEN message}

begin
  If id2=i and id1=FATHER
  then call TOKEN_RECEIVED
end;

MAIN1 {for use by node R-the root}

begin
  S(R):=first (LINKS).s
  termination_flag:=false;
  while not termination_flag do
    call TOKEN RECEIVED
    send TERMINATE(R)
    {will be received by nodes that have not yet terminated}
  end;

MAIN2 {for use by node i other than the root}

begin
  S(i):=first (LINKS).s
  termination_flag:=false; stop_run:=false;
  while not stop_run do wait for event;
end:
Figure 1 - Demonstration of chain-unfairness phenomenon

A number on link $l_{ij}$ refers to $\#l_{ij}$.

Figure 2 - A topology example for demonstration of the algorithm execution. Numbers in nodes correspond to identification numbers. Arrows between nodes represent the spanning tree. The numbers on links represent the weight (demand) of the corresponding link.
TABLE 1 - A single slot per link allocation from [2]

S(i) stands for a slot allocation for link \( l_{id,i} \) (id \( \rightarrow \) i's activation)

R(i) stands for a slot allocation for link \( l_{i,id} \) (i \( \rightarrow \) id activation).

<table>
<thead>
<tr>
<th>slot node-id</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S(2)</td>
<td>S(3)</td>
<td>R(2)</td>
<td>R(3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>R(1)</td>
<td>S(5)</td>
<td>S(1)</td>
<td>R(4)</td>
<td>S(4)</td>
<td>R(5)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>S(5)</td>
<td>R(1)</td>
<td>R(5)</td>
<td>S(1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>S(2)</td>
<td>R(2)</td>
<td></td>
<td>S(5)</td>
<td>R(5)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>R(3)</td>
<td>R(2)</td>
<td>S(3)</td>
<td></td>
<td>S(2)</td>
<td>R(4)</td>
<td>S(4)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TABLE 2 - The final allocation obtained from the DMF (or CMF) algorithm.

<table>
<thead>
<tr>
<th>slot node-id</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S(2)</td>
<td>S(3)</td>
<td>R(2)</td>
<td>R(3)</td>
<td>S(3)</td>
<td>R(3)</td>
<td>S(3)</td>
<td>R(3)</td>
<td>S(3)</td>
<td>R(3)</td>
</tr>
<tr>
<td>3</td>
<td>S(5)</td>
<td>R(1)</td>
<td>R(5)</td>
<td>S(1)</td>
<td>R(1)</td>
<td>S(1)</td>
<td>R(1)</td>
<td>S(1)</td>
<td>R(1)</td>
<td>S(1)</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>S(2)</td>
<td>R(2)</td>
<td></td>
<td>S(5)</td>
<td>R(5)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>R(3)</td>
<td>R(2)</td>
<td>S(3)</td>
<td></td>
<td>S(2)</td>
<td>R(4)</td>
<td>S(4)</td>
<td>R(2)</td>
<td>S(2)</td>
<td></td>
</tr>
</tbody>
</table>