Distributed on-line schedule adaptation for balanced slot allocation in wireless ad hoc networks

Theodoros Salonidis[†] and Leandros Tassiulas[†,‡]

[†] Department of Electrical and Computer Engineering and Institute for Systems Research
A.V. Williams Bldg. University of Maryland at College Park, MD 20742, USA
[‡] Department of Computer and Communication Engineering, University of Thessaly
Argonafton and Filellinon 38221 Volos, Greece

thsalon@glue.umd.edu, leandros@isr.umd.edu

***Abstract*—**

We propose an algorithm for design and on the fly modification of the schedule of a wireless ad hoc network for provision of fair service guarantees under topological changes. The primary objective is to derive a distributed coordination method for schedule construction and modification for any wireless ad-hoc network that operates under a schedule where transmissions at each slot are explicitly specified over a time period of length T.

We first introduce a fluid model of the system where the conflict avoidance requirements of neighboring links are relaxed while the aspect of local channel sharing is captured. In that model we propose an algorithm where the nodes asynchronously re-adjust the rates allocated to their adjacent links based only on local information. We prove that from any initial condition the algorithm finds the maxmin fair rate allocation in the fluid model. Hence, if the iteration is performed constantly the rate allocation will track the optimal even in regimes of constant topology changes.

Then we consider the slotted system and propose a modification method that applies directly on the slotted schedule, emulating the effect of the rate re-adjustment iteration of the fluid model. Through extensive experiments in networks with fixed and time varying topologies we show that the latter algorithm achieves balanced rate allocations in the actual slotted system that are very close to the maxmin fair rates. The experiments show also that the algorithm is very robust on topology variations, with very good tracking properties of the maxmin fair rate allocation.

I. INTRODUCTION

As wireless ad hoc networks evolve from the experimental to the commercial domain, there is a need for efficient bandwidth allocation of the scarce wireless resources to the network users. A major obstacle in this quest is the spatial contention of transmissions sharing the wireless medium. Spatial contention can be addressed either in the physical or MAC layer.

On one end, the physical layer uses only a single channel and wireless nodes transmit using a broadcast wireless medium. Then all links in a vicinity contend for use of this medium because a node's transmission reaches all others. This creates several versions of the problem of unintended broadcast transmissions (the most well known being the "hidden-terminal" and "exposed terminal" problems) and a family of random distributed MAC protocols ([22], [23]) to address them. Despite their distributed nature and flexibility, random access MAC protocols cannot offer deterministic bandwidth allocation guarantees.

Multi-channel wireless technologies address spatial contention at the physical layer where each channel is defined by a separate frequency or spread spectrum code. The idea is that if the links in a vicinity do not use the same channel, then conflict-free transmissions can take place at the same time. Even if this method mitigates collisions due to unintended broadcast transmissions, contention still exists because each wireless node is usually equipped with a single transceiver and cannot transmit or receive simultaneously. This form of contention necessitates coordination of node transmissions on channels and links by establishing conflict-free schedules [1]. According to such a schedule, each node can communicate to only one link at a time. Also, the endpoint nodes of each link must be coordinated to communicate during common time intervals. Any violation of the above two rules, results to a conflict. Conflict-free scheduling allows for explicit and guaranteed bandwidth allocation: the fraction of time a pair of nodes spends communicating conflict-free on a link determines the rate (bandwidth) allocated to this link.

Early work has indicated that finding perfectly conflict-free link schedules that satisfy a certain global optimal objective (such as minimum schedule length for a given set of link bandwidth allocation requirements) is a notoriously hard problem, even if global topology information is available [1][3]. The first distributed approach [2] started by flooding connectivity and traffic requirements in the entire network and then each node computed the conflict-free schedule by independently executing a centralized algorithm. This is clearly not efficient, especially when the network is dynamic.

The emergence of the Bluetooth multi-channel wireless technology [21] has inspired more refined research on distributed link scheduling schemes for Bluetooth ad hoc networks (termed as scatternets). These distributed techniques are divided in hard and soft coordination schemes. Hard coordination schemes [8] attempt to establish perfectly conflict-free link schedules. The advantage is that they can provide strict bandwidth allocation guarantees since no transmission conflicts exist. However, maintenance of the conflict-free property may come at the expense of significant communication overhead when there are dynamic changes in the network. On the other hand, soft coordination schemes [9][10][11] trade-off perfectly conflict-free transmissions for lower complexity and better robustness in dynamic network operation. The downside–lack of ability to provide bandwidth allocation guarantees.

We introduce a low complexity, ”hard-coordination” distributed algorithm that aims in establishing and maintaining maxmin fair bandwidth allocations in any slotted multi-channel wireless network. Maxmin fairness is an intuitive and desirable objective in application scenarios where no explicit knowledge exists about the bandwidth requirements of the users in the network. A maxmin fair allocation tries to allocate an equal amount of bandwidth to all users. If a user cannot utilize all the bandwidth because of a constraint, then the residual bandwidth is distributed to less constrained users. Among any feasible bandwidth allocations, a maxmin fair allocation ensures that the most constrained users are allotted the maximum possible bandwidth.

In this paper, we focus at the medium access layer and, as in [4][5][6] [7], we address fairness for single-hop flows (links) instead of multi-hop sessions. Two reasons motivate this approach. First, maintenance of end-to-end session state may not be possible in lightweight mobile nodes or even desirable in a highly mobile network. Still, transmissions must be coordinated such that robust and balanced access is provided to the higher layers. Second, provision of fairness on a multi-hop session basis can be viewed as an orthogonal objective. Recently, two distributed algorithms have been proposed in [31] and [32] for end-to-end utility-based fairness and maxmin fairness, respectively. Operating at a higher layer, these algorithms compute the fair session rates, but they do not enforce these rates–a distributed medium access mechanism is needed.

We first introduce a fluid model that captures only the bandwidth allocation constraints without taking into account the conflict-free requirement. In this model we propose a distributed algorithm that starts from an initial rate allocation and eventually converges to the maxmin fair solution after a series of asynchronous link rate adjustments. The slotted version of the algorithm attempts to emulate the one of the fluid model with the basic difference that whenever it adjusts the rate of a link it does so by re-assigning transmission slots directly on the network schedule without violating the conflict constraints. Since the fluid algorithm converges to the maxmin fair rates under asynchronous distributed operation, the slotted one is expected to have similar properties.

It should be noted that the maxmin fairness objective in slotted multi-channel wireless systems was first addressed in [7]. The authors provide an on-line scheduling policy and prove analytically that it converges to the maxmin fair solution. However, the policy uses global network information to compute the conflict-free link schedule and therefore cannot be implemented in practice. The slotted version of the distributed algorithm proposed here is implementable but there is no analytical proof for its exact convergence as in the fluid case. Through extensive simulations in static and dynamic networks we show that the algorithm possesses very good tracking properties of the maxmin fair rate allocation.

The rest of the paper is organized as follows. Section II presents the network model and definition of maxmin fairness. Section III introduces the fluid part of the asynchronous algorithm that computes the amount of rate adjustments. Section IV describes the scheduling technique that enforces these rate adjustments by means of conflict-free slot reallocations. In Section V the algorithm performance is evaluated. We discuss related work in Section VI. Section VII concludes.

II. NETWORK MODEL AND MAXMIN FAIRNESS DEFINITION

A. Network and communication model

The wireless ad hoc network is represented by a graph $G(N, L)$. An edge (i, j) signifies that nodes i and j are within range and have established a physical wireless link. Physical links are assigned communication channels such that there are no conflicts due to unintended broadcast transmissions. One way to achieve this is to associate every node with a unique channel; if each physical link is assigned the channel of one of the node endpoints, then, all transmissions between different node endpoints will occur in different channels. Bluetooth is a wireless technology that implements this method using spread spectrum signaling. Each node is associated with a unique frequency hopping (FH) sequence derived from its unique MAC address. Upon link establishment, one of the node endpoints is assigned as master and the other as slave. The link is assigned the FH sequence of the master. Although not orthogonal, Bluetooth FH sequences have been shown to perform well in practice [33]. Interference can be further mitigated using distributed assignment mechanisms that minimize the number of FH channels per locality[15][16][17]. In [13] it is shown that perfectly orthogonal access can be achieved if nodes within two wireless hops of each other are assigned orthogonal CDMA codes–if d is an upper bound on the intended physical links per node, a total of $2d(d-1)+2$ (instead of N) orthogonal channels are needed. References [13][14] propose distributed dynamic algorithms performing such assignments. Apart from using spread spectrum channels, interference can also be mitigated using directional antennae.

We will assume that one of the above techniques is used to avoid collisions due to unintended broadcast transmissions. However, each wireless node has a single radio transceiver and cannot communicate to more than one channel and link at a time (see Figure 1).

The system is slotted and all nodes are synchronized on a slot basis. Synchronization can be achieved using GPS clocks or signaling techniques similar to those employed in wired networks [12] modified for the wireless setting. Every system slot supports bidirectional transfer of data or control packets via a pair of equal duration half-duplex mini-slots. To implement conflict-free communications, each node i maintains a local periodic link schedule $\boldsymbol{S}_i$ of T (full-duplex) slots. In every slot of this schedule, a node can either communicate on a single link or remain idle. Communication on a link is conflict-free only if both endpoints have assigned concurrent slots on this link in their local schedules.

We use two models to represent bandwidth allocation. In the *slot model* the bandwidth allocated to a link l is expressed

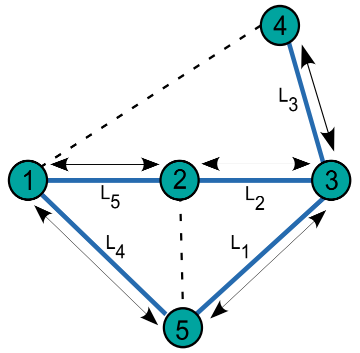


Fig. 1. Solid lines denote physical links (intended transmissions), while dotted lines denote wireless proximity. Links support bidirectional transfer per slot. Since links L_3 and L_5 use different channels, they can transmit simultaneously without conflict even if nodes 1 and 4 are within range. Still, every node can communicate to only a single link at a time due to the single radio transceiver constraint. Thus, only sets of links not having common node endpoints can transmit simultaneously without conflict (e.g. $\{L_3, L_4\}$ or $\{L_1, L_5\}$).

as the number of slots τ_l in a T-slot periodic conflict-free link schedule. The *fluid model* does not refer to a slotted system. The bandwidth allocated to a link l is the long-term fraction of time the node endpoints spend communicating conflict-free on this link. The two models serve different purposes. The fluid model is more general and intuitive and can be used to describe bandwidth sharing as well as notions such as feasibility and maxmin fairness. However a real system will always work in the discrete domain on a finite T-periodic schedule.

B. **Feasibility and maxmin fairness definition**

Under the fluid model, the *effective capacity* C_i of a node i, is defined as the maximum bandwidth a node provides its links for communication. If C_i is less than unity, then the node is always partially utilized by its adjacent links and remains idle for the rest of the time.

A link bandwidth allocation $R = (r_1, ..., r_l, ..., r_{|L|})$ is called *feasible* if there exists a conflict-free (not necessarily periodic) schedule that allocates to every link l, a long-term rate equal to r_l. The set of all feasible bandwidth allocation vectors defines the feasibility region, characterized by a set of constraints. Let $L(i)$ be the set of links sharing node i. Since i cannot communicate on different links simultaneously, an obvious constraint is that the sum of the rates of all links in $L(i)$ must be less than the node capacity. Interestingly, a node capacity of unity guarantees feasible bandwidth allocations only when the network topology is bipartite [1]. For a more general topology the characterization of the feasible region is not as straightforward. Still, according to [1] a node capacity equal to $2/3$ provides with a sufficient (albeit not necessary) characterization of feasibility. Summarizing the above, we reach the following node *capacity constraints* for feasibility:

$$\sum_{f \in L(i)} r_l \leq C_i \ , \ \ \forall i \in N \ , where$$

$$C_i = \begin{cases} 1 & \text{if } G(N, L) \ is \ bipartite \\ 2/3 & \text{otherwise} \end{cases} \tag{1}$$

How do nodes pick their effective capacity in a real distributed system? In multi-channel systems bipartite topologies can be formed by appropriate selection of the physical links to be formed. Such a selection is implicit in clustered architectures [18], [19], where each cluster (channel) is defined and controlled by a clusterhead node. Inter-cluster communication is performed by non-clusterhead gateway nodes that participate in multiple clusters. In such systems the network topology is by definition bipartite. Upon joining the network a node may query its neighbors whether a clustered architecture is used or not and set its effective capacity accordingly.

If a link l has a long-term arrival rate B_l we also need a *demand constraint* on the maximum allowable rate for this link:

$$r_l \leq B_l \tag{2}$$

A feasible rate allocation is maxmin fair (MMF) if the rate allocated to a link cannot be increased without hurting other links having equal or less rate. In Figure 1 the MMF allocation is $(r_{L_1}, r_{L_2}, r_{L_3}, r_{L_4}, r_{L_5}) = (1/3, 1/3, 1/3, 1/2, 1/2)$. We see that because node 3 is fully utilized, the rate of $1/3$ allocated to link L_1 cannot be increased without hurting the rates of the flows L_2,L_3 that share node 3 and have been allocated an equal rate. More formally, a rate allocation vector $\boldsymbol{r}$ is defined to be MMF if:

1) It is feasible i.e. satisfies the capacity and demand constraints given by eqs. (1) and (2).
2) It is lexicographically greater than any other feasible rate allocation vector $\boldsymbol{r}'$. This means that if we sort both $\boldsymbol{r}$ and $\boldsymbol{r}'$ by increasing order of their rates and we start comparing one by one the rates of the corresponding permuted vectors $\widetilde{\boldsymbol{r}}$ and $\widetilde{\boldsymbol{r}}'$ starting from the lowest index (which is 1), then after a possible set of equal rates there will be an index j such that $\widetilde{r_j}' < \widetilde{r_j}$.

Node i is defined as a *bottleneck node* to an adjacent link l if the sum of the rates of all links in $L(i)$ equals the node effective capacity C_i and the rate of link l is greater than or equal to the rate of all other links in $L(i)$. The definition of bottleneck node, gives rise to a distributed criterion to determine whether a given allocation is MMF or not:

MMF criterion: A bandwidth allocation is MMF if and only if every link l in the network satisfies at least one of the following conditions:

- The bandwidth allocated to link l equals its long-term arrival rate B_l.
- The link l has at least one bottleneck node.

For example, in Figure 1, we can easily verify that nodes 1 and 3 are bottleneck nodes for links L_4, L_5 and L_1, L_2, L_3 respectively.

The link MMF rates can be computed using an iterative, offline centralized algorithm. During each iteration, each node equally divides its available bandwidth to its adjacent links. The bottleneck nodes are the ones for which this division is minimum; the minimum ratio is the MMF rate for this iteration and is allocated to the links adjacent to the bottleneck nodes. We then remove the bottleneck nodes and their adjacent links from the network and reduce the available bandwidth of the remaining nodes by the amount consumed by the removed links. Any node whose available bandwidth becomes zero is also removed. In the next iteration, we consider the reduced network, determine the (next-level) bottleneck nodes and repeat the procedure. The process continues until all links have been allocated their rates. Upon termination, this algorithm yields the MMF link rates because the links removed in every iteration have at least one bottleneck node. The centralized algorithm is similar in spirit to the algorithm of Bertsekas and Gallager for wireline networks [29]–in our case, the resources to be shared are nodes instead of wired links and the entities sharing the bandwidth of the resources are wireless links instead of end-to-end sessions.

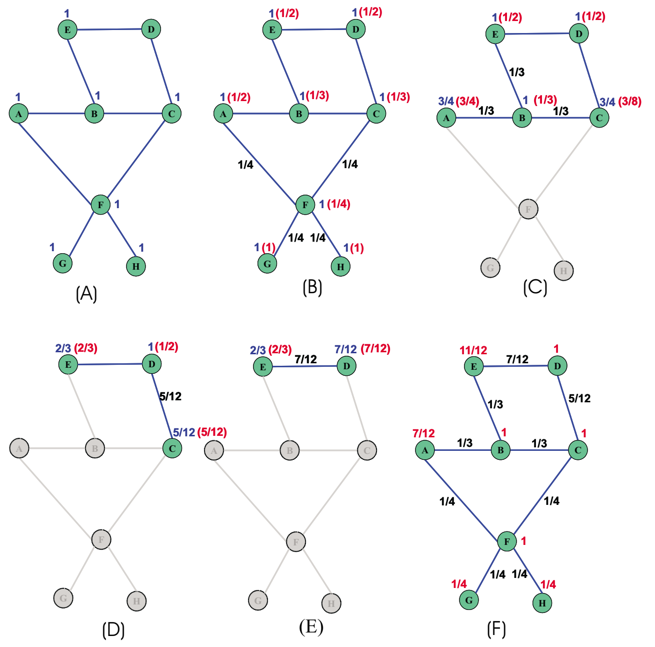


Fig. 2. (a) Initialization: All nodes set their effective capacities to 1 (bipartite topology) (b) Iteration 1: Bottleneck node is F–over all nodes, it provides the minimum fair share of 1/4 to its adjacent links. (c) Iteration 2: Bottleneck node is B (MMF rate is 1/3) (d) Iteration 3: Bottleneck node is C (MMF rate is 5/12) (e) Iteration 4: Bottleneck node is D, MMF rate is 7/12 (f) The MMF link rate allocation.

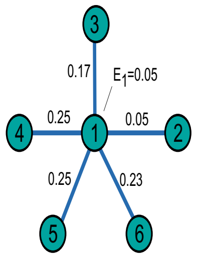
Figure 2 is an example of the centralized algorithm operation. The example illustrates that link-level max-min fairness is a global objective–the optimal allocation depends on the entire topology. Since nodes have access to only local information, they never know the MMF rates of their adjacent links. We seek an asynchronous distributed algorithm where nodes incrementally reach the global MMF link rate allocation via local rate adjustments. Such an algorithm would allow convergence to the MMF solution once the topology stabilizes for a sufficient amount of time. A second challenge, not addressed even by the centralized algorithm, is that the nodes must also reach a TDMA schedule that enforces these rates.

We first introduce an algorithm that computes the MMF rates using only local information. This algorithm is then used in the slotted system to guide slot re-assignments for rate adjustments. We thus target for rate computation and enforcement to occur in parallel. Our approach will be presented in detail in the following sections.

III. DISTRIBUTED ALGORITHM–FLUID MODEL

A. Fairness deficit

In this section, we introduce an asynchronous distributed algorithm for the fluid model that works in the feasible rates region and eventually converges to the MMF allocation. The algorithm starts from an arbitrary feasible link rate allocation $\boldsymbol{R}$. At asynchronous points in time each link is activated for a possible rate adjustment. The adjustment is such that at least one of the node endpoints becomes a bottleneck node for this link. A bottleneck node can be created if the node utilization equals C_i and the rate of the link increases so that it gets a rate greater than or equal to the rate of the other links adjacent to that node. The amount of this rate increase is called the *fairness deficit* and is computed by the fairness deficit computation (FDC) algorithm. The rate allocation of node i is the set $\boldsymbol{r}_i = \{r_{ij} : \; j \in N(i)\}$



step \ r	r_{12}	r_{13}	r_{14}	r_{15}	r_{16}	E_1	max_rate
0	0.05	0.17	0.25	0.25	0.23	0.05	0.25
1	0.10	0.17	0.25	0.25	0.23	0.00	0.25
2	0.20	0.17	0.20	0.20	0.23	0.00	0.23
3	0.215	0.17	0.20	0.20	0.215	0.00	0.215

Fig. 3. The FDC algorithm for link (1,2) by node 1. First, the available bandwidth $E_1 = C_1 - \sum_{j \in N(1)} r_{1j}$ of node 1 is given to link $(1,2)$. Then, at each step t, we consider the maximum rate set $M^{(t)}$ (denoted by the shaded entries). If r'_{12} does not belong in $M^{(t)}$, the total bandwidth of the links in $M^{(t)}$ and link $(1,2)$ is equally distributed to them. This process continues until link $(1,2)$ is in the maximum rate set. The last row is the new rate allocation $\boldsymbol{r}'_1$; the fairness deficit is $fd_{1\rightarrow 2} = 0.215 - 0.05 = 0.165$.

of rates allocated to its adjacent link flows. Starting from $\boldsymbol{r}_i$, the FDC algorithm for link (i,j) computes a new allocation $\boldsymbol{r}'_i$ in which the rate r'_{ij} belongs to the maximum link rate set. Then the fairness deficit $fd_{i\rightarrow j}$ of node i with respect to (i,j) is $r'_{ij} - r_{ij}$. Figure 3 is a representative example of the FDC algorithm operation; Figure 14(a) in Appendix B, contains the algorithm pseudocode, which includes the case when there is a demand constraint B_{ij} on link (i,j).

B. The fluid model algorithm

When a link (i,j) is asynchronously activated for rate adjustment, the following actions are performed:

1) Nodes i and j compute their fairness deficits $fd_{i\rightarrow j}$ and $fd_{j\rightarrow i}$ on link (i,j) and exchange their deficit values. The **link fairness deficit** is $fd_{ij} = min\{fd_{i\rightarrow j}, fd_{j\rightarrow i}\}$.
2) If the link fairness deficit is zero, then no rate adjustment takes place, steps 3 and 4 are not executed and no further action is taken.
3) If both deficits are non-zero, then the rate of link (i,j) is increased by fd_{ij}.
4) Nodes i and j adjust the rates of the rest of their adjacent links accordingly. If i is the minimum deficit node then its new link rate allocation is the one given by the FDC algorithm of $fd_{i\rightarrow j}$ in step 1. The maximum deficit node j reaches its new link rate allocation $\boldsymbol{r}'_j$ by applying again

the FDC on link (i,j) with an upper bound constraint of $r_{ij}+fd_{ij}$.

Note that in order to do the above adjustments we only need to reduce the rates of certain links adjacent to nodes i and j except link (i,j) the rate of which is increased by fd_{ij}.

Convergence Theorem: Given a static topology and an arbitrary initial feasible network rate allocation $\boldsymbol{R}$, the above algrotihm converges to the network maxmin fair solution after a finite number of link activations for rate adjustment.

Proof: See Appendix A.

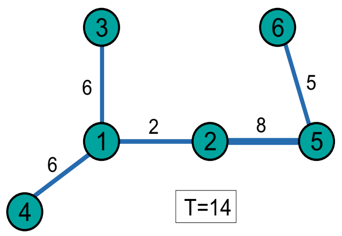
The algorithm is self-terminating–no explicit message needs to be sent to the entire network to signal convergence. When a link is activated for a possible rate adjustment, adjustment occurs only if the link fairness deficit is non-zero. Upon convergence, all links will have at least one bottleneck node and the link fairness deficit will be zero for all links.

IV. DISTRIBUTED ALGORITHM–SLOT MODEL

A. Fairness deficit computation and slot assignment algorithm

The fluid algorithm guarantees convergence to the network maxmin fair rates but does not yield a conflict-free schedule that realizes these rates. This is because the fluid model does not refer to a slotted system but is mainly concerned on how to redistribute the bandwidth.

The slotted algorithm emulates the one of the fluid model with the basic difference that whenever it adjusts the rate of a link it does so by re-assigning transmission slots directly on the network schedule without violating the conflict constraints. Since the fluid algorithm converges to the maxmin fair rates under asynchronous distributed operation, the slotted one will have similar properties provided it yields a conflict-free schedule after each rate adjustment. The slotted fairness deficit computation algorithm for node i, uses the one of the fluid model to reach from discrete slot allocation $\boldsymbol{\tau}_i$ to $\boldsymbol{\tau}'_i$, and outputs the difference vector $\boldsymbol{x}_i=\boldsymbol{\tau}'_i-\boldsymbol{\tau}_i$. An example of the detailed operation of the slotted FDC is shown in Figure 5.



slot# id	0	1	2	3	4	5	6	7	8	9	10	11	12	13
S_1	4	3	3	4	3	4	3	4	2	3	2	4	3	4
S_2	-	5	5	5	5	5	5	5	1	5	1	-	-	-
S_3	-	1	1	-	1	-	1	-	-	1	-	-	1	-
S_4	1	-	-	1	-	1	-	1	-	-	-	1	-	1
S_5	6	2	2	2	2	2	2	2	6	2	-	6	6	6
S_6	5	-	-	-	-	-	-	-	5	-	-	5	5	5

Fig. 4. A wireless ad hoc network using a $T=14$ periodic conflict-free schedule. The link weights denote the number of conflict-free slots allocated to each link. Each slot entry j in the local schedule $\boldsymbol{S}_i$ of node i, signifies that node i has assigned this slot to link (i,j).

step		(1,2)	(1,3)	(1,4)	rem	Actions
0	τ_1	2	6	6	0	T=14
1	r_1	2/14	6/14	6/14	0/14	$r_{1j}=\tau_{1j}/T$
2	r_1	0.333	0.333	0.333	0.000	fluid FDC
3	τ_1	4	4	4	2	$\tau_{1j}=\lfloor r_{1j}T \rfloor$
4	τ_1	6	4	4	0	Give remainder slots to (1,2)
5	x_1	+4	-2	-2	0	$x_{1j}=\tau_{1j}-\tau_{1j}$

Fig. 5. The slotted FDC for node 1 on link $(1,2)$ in the network of Fig. 4: (1) slots are converted to normalized rates (2) fluid model FDC is applied to rates. (3) resulting rates are ”quantized” to slots. (4) The excess slots due to the quantization of step 3 are given to link $(1,2)$. (5) The resulting rate difference vector $\boldsymbol{x}_1$. The fairness deficit for link $(1,2)$ is 4 slots.

Given $\boldsymbol{x}_i$, a positive (negative) element x_{ik} means that the rate of link (i,k) must be increased (decreased) by x_{ik} slots. A zero element indicates no change in the rate of the corresponding link. The set of surplus links (i.e. the links affected by the rate adjustment on link (i,j)) is $X_i^-=\{(i,k):x_{ik}<0\}$. Also x_{ij} is positive and equal to the fairness deficit amount of slots that must be assigned to link (i,j).

The slot assignment algorithm decides for each surplus link (i,k) **which** x_{ik} out of the τ_{ik} current slot positions will be re-assigned to link (i,j). To maintain the conflict-free property, both endpoint nodes must eventually assign to (i,j) the same slot positions in their link schedules.

The slot assignment algorithm consists of two phases. In **Phase I**, node i takes into account the link schedule of j and assigns slot positions to link (i,j) in the following prioritized way:

1) First, link (i,j) is assigned slot positions that are currently idle in both link schedules $\boldsymbol{S}_i$ and $\boldsymbol{S}_j$, if such positions exist.
2) If step 1 did not find enough matching slot positions, link (i,j) is assigned slot positions where j is idle and i currently uses for a surplus link (i,k), if such positions exist.

The number of slot positions that matched during phase I may still be less than the required deficit for link (i,j). For each surplus link (i,k) that Phase I selected only m_{ik} out of x_{ik} slots, **Phase II** randomly selects extra $x_{ik}-m_{ik}$ slot positions that are still assigned to (i,k) in $\boldsymbol{S}_i$ and reassigns them to link (i,j). The algorithm outputs a list indicating the extra slot positions that should be assigned to link (i,j). The algorithm pseudocode can be found in Figure 14(b) of Appendix B.

As an example of the slot assignment operation, node 1 is called to decide on the extra slot positions that will be assigned to link $(1,2)$ based on its own and node 2's local schedules (Figure 6). The rate difference vector (row 5 in Figure 5) indicates that links $(1,3)$ and $(1,4)$ must give away two slots each and link $(1,2)$ should be assigned four extra slots. By matching the idle slots of $\boldsymbol{S}_2$, node 1 reassigns slot positions $\{7,12\}$ from $(1,3)$ and $\{11,13\}$ (selected randomly from $\{0,11,13\}$) from $(1,4)$ to link $(1,2)$.

slot#	0	1	2	3	4	5	6	7	8	9	10	11	12	13
S_1	4	3	3	4	3	4	3	4	2	3	2	4	3	4
S_2	-	5	5	5	5	5	5	5	1	5	1	-	-	-

Fig. 6. Idle slot positions $\{12\}$ and $\{0, 11, 13\}$ of $\boldsymbol{S}_2$ match with ones assigned to links $(1, 3)$ and $(1, 4)$ in $\boldsymbol{S}_1$ respectively. Link $(1, 2)$ is finally assigned slot positions $\{7, 11, 12, 13\}$.

B. Signaling schedule updates

After the slot assignment algorithm, the rate increase on a link decreases the rates of some of the other links adjacent both endpoint nodes. To maintain the conflict-free schedule property, the affected nodes must be notified to update their own local schedules to reflect this change. A *schedule update* (SC) control packet sent from node i to node j informs its recipient how it should modify its local schedule and contains the following information:

- A field specifying if the packet concerns a rate increase (SC_inc) or decrease (SC_dec) SC packet (1 bit).
- A list of slot positions that need to be modified in the recipient's local schedule (T-bits). For an SC_inc packet the indicated positions will be assigned to link (i, j) in the updated local schedule of recipient node j, while for an SC_dec packet they will be assigned as idle.
- The number of slots the recipient should wait before applying the above schedule update ($\lceil log_2 T \rceil$ bits).

Also, all nodes participating in the link rate adjustment must know when to update their local schedules. Starting from slot s where the link was activated for rate adjustment, the commit slot offset is the number of slots needed for the schedule update to be propagated to all the affected nodes in the one-hop neighborhood of link (i, j). The **commit slot offset** $coff_{ij}^{(s)}$ is locally computed on slot s and is appropriately included in the SC control packets to let each node know when it should apply the update. After $coff_{ij}^{(s)}$ slots, the last node receives an SC packet and all affected nodes (including nodes i and j) apply the schedule update starting on the next slot.

C. The commit slot offset computation

Given a node i and a slot s in its current periodic schedule, the *multicast slot offset* $b_i^{(s)}(M(i))$ on the neighbor subset $M(i)$ of $N(i)$, is the number of slots needed by i to communicate with all nodes in $M(i)$ starting from slot s. After node i performs the slot assignment algorithm, it needs $A_i^{(s)} = b_i^{(s)}(N(i))$ slots to send the schedule update to all its neighbors. The other end node j receives the update after $\alpha = b_i^{(s)}(\{j\})$ slots and according to its own schedule $\boldsymbol{S}_j$, it needs additional $b_j^{(s+\alpha)}(N(j) - \{i\})$ slots to update the rest of its neighbors. Therefore starting from slot s, node j will need a total of $B_j^{(s)} = \alpha + b_j^{(s+\alpha)}(N(j) - \{i\})$ slots to propagate the schedule update. The commit slot offset is the number of slots after slot s until both i and j reach all their neighbors: $coff_{ij}^{(s)} = max\{A_i^{(s)}, B_j^{(s)}\}$. Referring to Figure 6, assume that node 1 has just performed the slot assignment algorithm at slot $s = 8$. Given $\boldsymbol{S}_1$, node 1 will need $A_1^{(8)} = b_1^{(8)}(\{2, 3, 4\}) = 3$ slots to send the schedule update. Node 2 will receive the schedule update at slot 10, and according to $\boldsymbol{S}_2$ it will need additional $b_2^{(10)}(\{5\}) = 5$ slots to reach node 5 (on slot 1 of its periodic schedule). Thus, starting from slot $s = 8$ node 2 will need $A_1^{(8)} = 2 + 5 = 7$ slots for the schedule update propagation and finally the commit slot offset is $coff_{12}^{(s)} = max\{A_1^{(s)}, B_2^{(s)}\} = 7$ slots.

D. The complete algorithm

When a link (i, j) is activated for rate adjustment at slot s, the following actions are performed:

1) Nodes i and j compute their (discrete) fairness deficits $fd_{i \to j}$ and $fd_{j \to i}$ on link (i, j) and exchange two *fairness deficit* (FD) control packets. The FD packet sent by each node n contains the following information:
   - The node's calculated discrete fairness deficit with respect to link (i, j) ($\lceil log_2 T \rceil$ bits).
   - The number of slots $B_n^{(s)}$ node n needs to propagate the schedule update to all its neighbors in case it turns out to be the maximum deficit node ($\lceil log_2 T \rceil$ bits).
   - A T-bit vector $\boldsymbol{I}_n$ indicating the idle slot positions in its local schedule $\boldsymbol{S}_n$ (T bits).
2) If any of the two fairness deficits is zero, no rate adjustment takes place, the rest of the steps are not executed and no further action is taken.
3) If both deficits are non-zero, then the rate of link (i, j) must be increased by the minimum of the two fairness deficits. The minimum deficit node is the one with the smaller deficit or in the case of equal deficits the one with smaller id.
4) If i is the minimum deficit node, then based on the FD packet received by j:
   - Given $\boldsymbol{I}_j$, it executes the slot assignment algorithm to determine the list of extra slot positions that will be assigned to link (i, j).
   - It computes the number of slots $A_i^{(s)} = b_i^{(s)}(N(i))$ it needs to propagate the schedule update to all its neighbors. The commit slot offset is then $coff_{ij}^{(s)} = max\{A_i^{(s)}, B_j^{(s)}\}$.
5) Then i sends j an SC_inc packet with the list of slot positions decided by the slot assignment algorithm for link (i, j), and an SC_dec packet to the rest of its neighbors to notify them when and which slots of their local schedules they should set as idle. As soon as j receives the SC_inc packet, it sends an SC_dec packet to all its neighbors similar to node i.
6) At (global) time instant $s + coff_{ij}^{(s)}$, node i, node j and all their one-hop neighbors apply the change they received earlier in the SC packets; the schedule adjustment is complete.

E. Simultaneous link activations for rate adjustment

In this section we provide the additional mechanisms needed to maintain the conflict-free property of the local schedules when multiple links are adjusting their rates simultaneously.

Links can be asynchronously and independently activated for rate adjustment on the slots assigned to them for communication by the current network conflict-free schedule. According to this schedule the links that can transmit simultaneously do not have common node endpoints. Therefore only such links are activated for rate adjustment (increase) on the same slot.

If link (i, j) is activated for a rate adjustment on slot s, the interval $\{s, s+1, ..., s+coff_{ij}^{(s)}\}$ until the endpoints update their local schedules is defined as their **busy period**. To maintain the schedule conflict-free property, no adjacent link to the endpoint nodes i and j must be activated for rate adjustment in any slot within the busy period. Thus, during their busy period, nodes i and j do not initiate or respond to requests (i.e. FD packets) for rate adjustment from their neighbors. If a neighbor node happens to initiate and does not receive a response, it retries again after a random number of slots. After the busy period the endpoints i and j become again available for rate adjustment with other neighbors.

Also, during the busy period nodes i and j lock the extra slot positions decided by the slot assignment algorithm for link (i, j) until the end of their busy period where they update their schedules. If during this time they receive an SC_dec packet from any of their neighbors k that was produced by a simultaneous rate increase of link (k, l), they set as idle only the indicated slots that do not coincide with the locked ones. This prevents one node endpoint overwriting the extra slot positions given to (i, j) if the SC_dec packet happens to indicate common positions to be set as idle.

F. Protocol communication Requirements

The amount of control information needed by the protocol depends only in the system period T and not in the network dimensions such as size or density. The FD and SC packets consist of $2\lceil log_2 T\rceil + T$ bits and $1 + T + \lceil log_2 T\rceil$ bits respectively. Thus the protocol requirement in bits per control packet is:

$$B_{control} = 2\lceil log_2 T\rceil + T \ \ bits \tag{3}$$

Since the control and data packets share the same slots, this sets the minimum (excluding FEC, headers etc) half-duplex minislot size in the system. If the radio transmission rate is R_{tx} bits/sec, the minimum duration of a single slot system packet is $2(2\lceil log_2 T\rceil + T)/R_{tx}$ sec. Higher transmission rates allow for shorter slot durations for a fixed T or larger schedule periods T for a fixed slot size.

V. PERFORMANCE EVALUATION

A. Experimental model and setting

We have implemented a packet-level simulator environment in C++ for evaluating the algorithm performance. The simulator includes the generation of various static and dynamic topology scenarios as well as an implementation of the proposed protocol.

Topology dynamics are modeled by having links going up and down in a static baseline topology [28]. This model captures the way mobility is manifested in multi-channel systems without delving into the details of the complex hand-off and link establishment protocols that should be used by a multi-channel system when nodes actually move. While important, such protocols are out of the scope of this paper. Also this model allows for explicit control of parameters that affect the protocol performance such as topology density and frequency of topology changes.

Each link in the baseline topology cycles independently between an ACTIVE (link "up") and INACTIVE ("link down") state. A link remains ACTIVE for a geometrically distributed number of slots with mean T_{active}. Since all links alternate between the two states independently, the long-term fraction of time p a link is ACTIVE equals the average percentage of active links in the baseline topology at any time. In addition, certain multi-channel technologies impose a limit on the number of physical links a wireless node can maintain simultaneously. This restricts the maximum node degree to D_{max} (e.g. in Bluetooth D_{max} is 7). The parameter T_{active} is used to tune the rate of topology changes while p and D_{max} affect the average network density. The frequency of rate adjustments is controlled by the protocol parameter T_{adjust}. After a link rate adjustment, the endpoint nodes agree on a random rate adjustment timer chosen uniformly between 0 and T_{adjust} slots. The timer decreases on each future time slot the link is used for transmissions. When the timer expires, the link is activated for rate adjustment.

We use two metrics for the algorithm performance evaluation:

- **Relative computation error**: If the MMF rate of a link (i, j) at time t is $r_{ij}^{MMF}(t)$ and the computed rate is $r_{ij}(t)$, the relative computation error for link (i,j) at time t is $|1 - r_{ij}(t)/r_{ij}^{MMF}(t)|$. For each slot t, we consider the maximum and average relative computation error over all currently ACTIVE links. After each topology change, the reference link MMF rates are computed off-line using the centralized algorithm.
- **Control Overhead**: During network operation, a slot can be idle, used for transmission of a DATA packet or exchange of control information conveyed by the FD and SC packets. The control overhead is the ratio of control packets over the total number of packets transmitted during a simulation run.

In the following experiments we consider bipartite topologies because the entire feasibility region is can be captured by local conditions in this case (i.e. if every node sets $C_i = 1$ in equation (1) feasibility is guaranteed). As mentioned earlier, bipartite topologies arise in clustered architectures [18], [19] or in some instances of Bluetooth-based ad hoc networks [21]. In the non-bipartite case, nodes can set their effective capacity to $2/3$ to guarantee feasible rate allocations; the algorithm will yield MMF rates with respect to this fractional capacity.

We use an $N = 100$ node bipartite baseline topology with 50 nodes per bipartite set. This yields a rich set of $N^2/4 = 2500$ possible links in the baseline topology that can be ACTIVE or INACTIVE at any time. In terms of traffic demands, all links are assumed backlogged (no demand constraints) when ACTIVE.

B. Experiments on static networks

Given the baseline topology, the parameters p and D_{max} are used to derive static topologies of various density and maximum degree characteristics (e.g. Figure 7). All simulations in static topologies were run for 500000 slots.

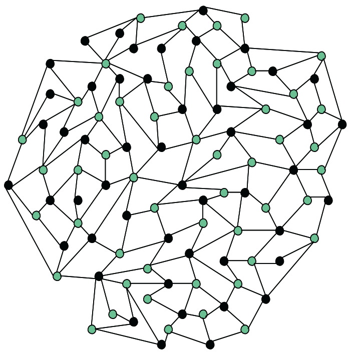


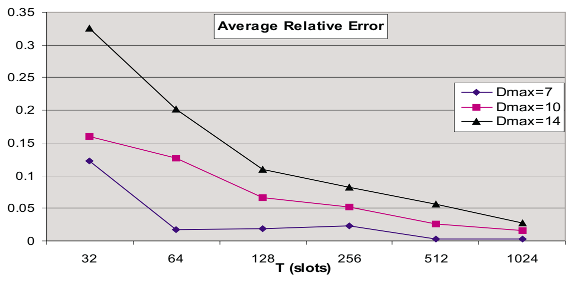
Fig. 7. A sample $N = 100(50/50)$ bipartite topology of $p = 0.1$ and $D_{max} = 7$ derived from the baseline topology graph. Only ACTIVE links are shown.

In Figures 8 and 9 we set $p = 1.0$ so that every node has a degree of D_{max}. The target MMF rate every link in the network must reach is $1/D_{max}$ (approximated by T/D_{max} slots). Figure 8 shows the effect of the schedule period T and maximum degree constraint D_{max} on the average and maximum relative errors. For a fixed D_{max}, both errors decrease as T increases. One reason is that a larger period provides a better approximation to the reference (continuous) MMF rates.

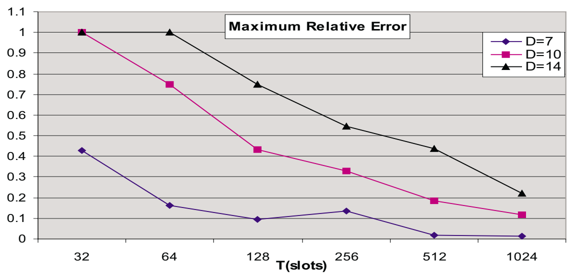
For example a period of $T = 64$ cannot provide enough granularity for a $D_{max} = 14$ and the resulting errors are very high. The other reason is that a larger T offers more transmission slots to a link per period. This incurs more frequent expirations of the rate adjustment timer, and hence more overall activations for link rate adjustment. This is also the explanation for the increase in the control overhead in Figure 9 as the period T increases.

The maximum node degree D_{max} has a more pronounced effect both in errors and control overhead. This is illustrated by the distance between the curves in both Figures 8 and 9. In the error curves, the effect of D_{max} decreases as the period T increases. After $T = 1024$ slots, the average relative error becomes less than 3% and the maximum error less than 20% for all cases. However, in terms of control overhead, the difference between the curves does not decrease with T. Thus for $T = 1024$, a $D_{max} = 7$ spends only 3% of transmissions in exchange of control packets while a $D_{max} = 14$ spends 17%. To keep the control overhead low, we need to reduce the frequency of rate adjustments that is controlled by the T_{adjust} parameter.

Figure 10(a) illustrates the effect of T_{adjust} on a ($T = 1024$, $D_{max} = 14$) system. By increasing T_{adjust} (hence decreasing the frequency of link rate adjustments) the control overhead decreases without any noticeable effect in the resulting maximum and average discrepancy from the MMF solution. At $T_{adjust} = 16384$ slots, the control overhead becomes negligible. Still, decreasing the frequency of link activations leads to a slower convergence. This will become obvious in the experiments of the dynamic topologies.



(a) Average relative error



(b) Maximum relative error

Fig. 8. (a) Average and (b) Maximum Relative Errors for a static network of $N = 100$ $p = 1.0$ and $T_{adjust} = 512$ slots for various choices of T and D_{max}. The average and maximum relative errors are computed over all active links at the last slot of the simulation.

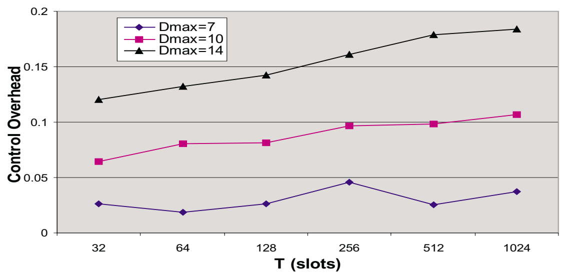
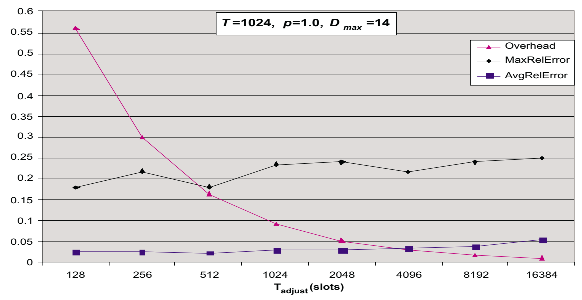


Fig. 9. Control Overhead for a static network of $N = 100$, $p = 1.0$ and $T_{adjust} = 512$ slots for various choices of T and D_{max}.

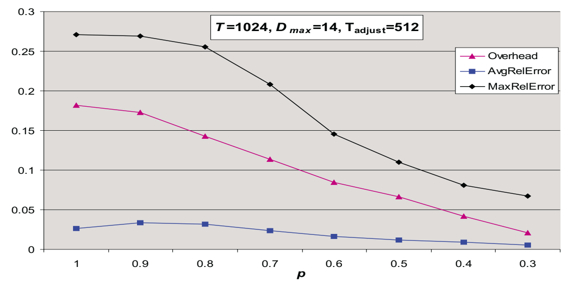
Figure 10(b) shows the effect of the topology density parameter p on the three metrics of interest. As the density decreases, less nodes need to establish the maximum number of links D_{max} and this leads to a reduction of both errors and control overhead in the network.

C. Experiments on dynamic networks

The parameter controlling the network dynamics is T_{active} for the rate of topology changes. To see how the time scale of topology dynamics affects the algorithm performance, we use as system technology parameters the ones of Bluetooth. Bluetooth supports a raw transmission rate of $R_{tx} = 1Mbps$ and a maximum number of simultaneously active physical links $D_{max} = 7$. The system slot duration is $1.25ms$. We use a pe-

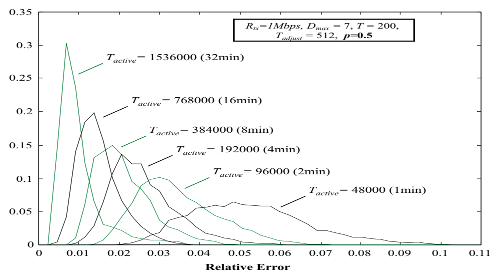


(a) Effect of reduction in the frequency of link rate adjustments.

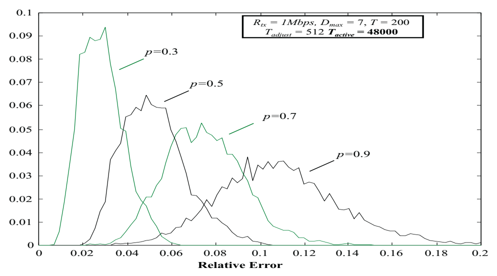


(b) Effect of the density topology parameter p.

Fig. 10. (a) Effect of reduction in the frequency of link rate adjustments. (b) Effect of the density topology parameter p.



(a) Effect of rate of topology changes.



(b) Effect of topology density.

Fig. 11. Effect of (a) rate of topology changes and (b) topology density in the distribution of the average relative error.

riod of $T = 200$ slots, which is the maximum that can be supported by the current Bluetooth specification [1]. All simulations were run for 500000 slots. We consider the pdf distribution of the average relative error during the last 100000 slots.

Figure 11 illustrates the effect of mobility and network density on the error distribution. The bell-shaped curves indicate that the MMF rate discrepancy experienced by an average link generally oscillates around a mean value. In Figure 11a, we let a link spend an equal average amount of time in the ACTIVE or INACTIVE state, by setting $p = 0.5$. The average time T_{active} a link alternates between the two states varies from $32min$ (1536000 slots) to $1min$ (48000 slots). As the rate of topology changes increases, both error mean and variance increase. This is illustrated by a right-shift and "spreading" of the error distribution curves as the parameter T_{active} decreases. For a quasi-static network ($T_{active} = 32min$), the MMF discrepancy of an average link is centered at 0.7% and varies between 0.2% and 4%. For $T_{active} = 1min$ the peak consists of a range of error values ($4\% - 6\%$) and the overall error dynamic range is $2\% - 10\%$.

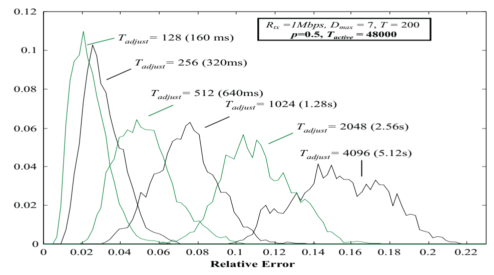
For the same rate of topology changes, the mean and variance of the average relative error increase with topology density (Figure 11b). The reason is that a denser topology allows for less simultaneous conflict-free transmissions per period and hence less frequent expirations of the rate adjustment timer per link. Therefore rate adjustments are happening at a slower rate and this affects the ability of the algorithm to track topology changes. Still, even in the most dense topology ($p = 0.9$) and high rate of topology changes of $T_{active} = 1min$ (48000 slots), an average link will achieve above 80% of its target MMF rate.

Figure 12 shows the effect of the rate adjustment parameter T_{adjust} in the most dynamic case where links form and fail every 1 minute (48000 slots) on the average. As T_{adjust} varies from $5.12s$ (4096 slots) to $160ms$ (128 slots), the error mean and variance decrease slightly (Figure 12a) but the control overhead increases (Figure 12b). For $T_{adjust} = 160ms$ (128 slots), the error is centered at 2% of the MMF rate but the control overhead needed to sustain it amounts to 27% of the overall number of transmissions. A T_{adjust} greater than 640ms (512 slots) keeps the overhead below 9% but the error mean and variance will gracefully increase according to Figure 12a.

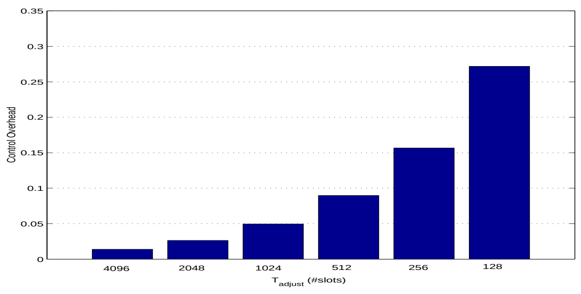
Figure 13 illustrates how topology dynamics and density affect the algorithm performance had the reference technology specification allowed for a larger D_{max}. The curve trends are the same as in Figure 11 but the error means and variances increase with D_{max}. This shows the algorithm performance degradation for technologies using a certain radio transmission rate and wish to support a larger maximum number of MMF links per node in a dynamic network.

Technologies supporting higher transmission rates result in

[1]Half duplex mini-slots in our model correspond to single-slot Bluetooth baseband ACL packets. The payload size of these packets is limited to 240 bits. If we exclude the higher layer headers and the CRC, only 216 bits are left for the protocol information (DH1 packets). When FEC is added (DM1 packets), the available space goes down to 136 bits. Using equation (3), we can see that the maximum period T for DH1 packets is 200 slots and for DM1 packets 122 slots.

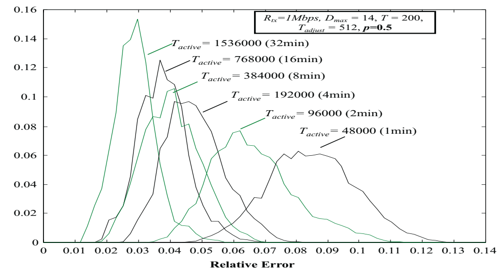


(a) Effect on T_{adjust} on average relative error

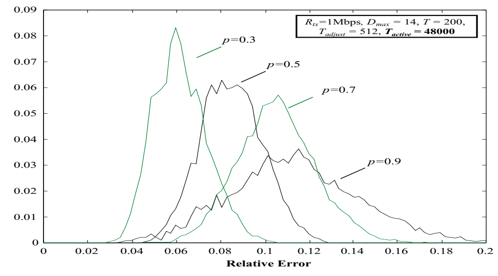


(b) Effect on T_{adjust} on control overhead.

Fig. 12. Effect of frequency of link activations on (a) the average relative error and (b) control overhead.



(a) Effect of rate of topology changes in the average relative error distribution.



(b) Effect of topology density in the average relative error distribution.

Fig. 13. $D_{max} = 14$: Effect of (a) rate of topology changes and (b) topology density in the distribution of the average relative error.

a better performance because they can use a shorter slot duration. For example if $R_{tx} = 2Mbps$ in the reference system, the system slot duration is $0.625ms$ instead of $1.25ms$ and therefore "$T_{active} = 2min$" in Figure 11a will now correspond to the error distribution of $T_{active} = 192000$ instead of the one of 96000 slots. As we double the transmission rate, we can see the corresponding performance improvement by moving one error distribution curve to the left in Figures 11a, 12a, 13a and one point to the left in Figure 12b for the control overhead.

VI. RELATED WORK

The maxmin fairness objective has been addressed for both single channel and multi-channel wireless systems. Fairness is defined and addressed for single-hop flows in all cases. Single channel systems are considered in [4][5][6]. The work in [4] uses a weighted fairness scheme to first allocate a minimum fair bandwidth to the network flows and then maximize the system utilization subject to this allocation. This approach can reach the maxmin fair allocation using appropriate flow weights. However, the weight computation would require knowledge of the MMF rates. This in turn would require a global network MMF rate pre-computation phase a difficult task in a large dynamic network. Nandagopal et al. [6] define fairness in terms of maximizing total logarithmic user utility functions and implement proportional fairness within this framework. Maxmin fairness is mentioned as a asymptotic case of this utility fairness model. A centralized and a distributed algorithm specifically targeted for maxmin fairness are proposed in [5]. The centralized algorithm reaches an approximate solution for large networks because it relies on the computation of the clique corpus of a graph, which is an NP-complete problem. In the distributed algorithm a node maintains a subset of the contention graph and heuristically computes a coarser allocation.

It should be noted that in [4][5][6], the distributed algorithms that approximate the fairness models are implemented using a random access MAC protocol and attempt to achieve the desired rates by setting a per-flow back-off timer according to the fair weight of the flow. Since random access cannot support strict bandwidth allocation guarantees fairness can be achieved only in a probabilistic sense in this case (very large time scales).

The work in [7] defines the maxmin fairness objective in a slotted multi-channel system using scheduled access and provides a scheduling policy that achieves maxmin fair allocation of flows. At each slot, a node first assigns appropriate weights to each of its adjacent flows by using a round robin token generation scheme. Then the flows that constitute a maximum weighted matching on the network are scheduled to transmit conflict-free. This step makes this approach unsuitable for distributed implementation because it requires global topology information for computing the maximum weighted matching.

DSSA [8], a distributed TDMA scheduling algorithm for Bluetooth scatternets, cannot be applied to the maxmin fairness objective. In DSSA nodes start with an knowledge of demands on their adjacent links and try to reach a conflict-free schedule of short length that satisfies these demands. However maxmin fairness is a global objective. Hence, to use this algorithm one must first pre-compute the MMF rates and then provide them as local traffic demands to every node in the network, something not practical.

Distributed algorithms for MMF rate computation for multi-hop sessions have also been studied extensively in the wireline networks context [24][25]. Our algorithm is similar by being asynchronous, distributed and targeting the MMF rates. The difference is that these algorithms perform only the fluid model part: they only compute the MMF rates but do not specify how to enforce them. The problem of enforcing the rates is treated separately by using end-to-end or hop-by-hop link schedulers and traffic shapers [26][27]. This separation is perfectly justified in the wireline networks context due to the link scheduling independence. In the wireless case, a rate adjustment on a link has an effect on the rates of links adjacent to both endpoint nodes and the problems of rate computation (fairness deficit computation) and rate enforcement (conflict-free slot assignment) must be addressed jointly.

VII. Conclusions

Future deployment of wireless ad hoc networks calls for decentralized techniques that allocate efficiently the scarce wireless medium to mobile users. We presented a distributed asynchronous algorithm of low complexity aiming for maxmin fairness. Bandwidth allocations are realized by conflict-free periodic link schedules. This implies both short-term (to the extent of the period T) and long-term fairness properties.

A unique feature of the distributed scheduling technique is that it does not assume any initial knowledge about the (global) MMF objective. Instead, the rate computation and enforcement occur simultaneously by means of local and incremental conflict-free schedule updates. This incremental property allows for natural adaptation to network dynamics without the need to suspend communications and restart the schedule computation from scratch. The scheduling mechanism is driven by the rate computation algorithm, which converges to the MMF solution under the fluid model. Still, when emulating the fluid algorithm in the slotted world the convergence is not exact and there are restrictions and trade-offs a designer has to take into account. To this end, we provide an analysis of the algorithm communication requirements and its effect on the design choices of a technology supporting it.

The algorithm was extensively tested under various technology choices and topology dynamics. For static networks it demonstrated excellent convergence properties especially as the schedule period T increases. For dynamic scenarios, an average link typically experiences a certain mean MMF discrepancy with a finite variance. Performance gracefully degrades with the increase in the rate of topology changes, network density and desired maximum number of physical links supported by a wireless node. In highly dynamic scenarios and stringent technology constraints (modest R_{tx} and high D_{max}), the incremental nature of the algorithm allows the network to be reasonably close to the maxmin fair solution most of the time. In addition, the frequency of link rate adjustments can be fine-tuned to get acceptable performance for low control overhead.

References

[1] B. Hajek and G. Sasaki, *Link Scheduling in Polynomial Time*. IEEE Trans. Inform. Theory, No 5, Vol. 34, 1988.

[2] M. J. Post, A. Kershenbaum and P.E. Sarachik, *A Distributed Evolutionary Algorithm for Reorganizing Network Communications*. Proc. MILCOM'85, Boston, MA, Oct. 1985.

[3] M. Post, P. Sarachik and A Kershenbaum, *A Biased Greedy Algorithm for Scheduling Multihop Radio Networks*. 19th Annu. Conf. on Information Sciences and Systems, Johns Hopkins Univ., March 1985.

[4] H.Luo, S. Lu and V. Bharghavan, *A new model for packet scheduling in multihop wireless neworks*. Proceedings of ACM MobiCom 2000, Boston MA, August 2000.

[5] X.L. Huang, B. Bensaou, *On maxmin Fairness and Scheduling in Wireless Ad-Hoc Networks: Analytical Framework and Implementation*. Proceedings of IEEE/ACM MobiHoc, Long Beach CA, Oct. 2001.

[6] T. Nandagopal, T. Kim, X. Gao and V. Bharghavan, *Achieving MAC layer fairness in Wireless Packet Networks*. Proceedings of ACM MobiCom 2000, Boston MA, August 2000.

[7] L. Tassiulas and S. Sarkar, *Maxmin Fair Scheduling in Wireless Networks*. Proceedings of Infocom 2002, New York, 2002.

[8] N. Johansson, U. Korner, L. Tassiulas, *A distributed scheduling algorithm for a Bluetooth scatternet*. Proc. Of the 17th International Teletraffic Congress, ITC '17, Salvador da Bahia, Brazil, Sep. 2001.

[9] A. Racz, G. Miklos, F. Kubinszky, A. Valko, *A Pseudo Random Coordinated Scheduling algorithm for Bluetooth Scatternets*. Proceedings of IEEE/ACM MobiHoc, Long Beach CA, Oct. 2001.

[10] N.Johansson, F. Alriksson, U. Jonsson, *JUMP mode - a dynamic window-based scheduling framework for Bluetooth scatternets*. Proceedings of IEEE/ACM MobiHoc, Long Beach CA, Oct. 2001.

[11] Simon Baatz, Matthias Frank, Carmen K uhl, Peter Martini, Christoph Scholz, *Bluetooth Scatternets: An Enhanced Adaptive Scheduling Scheme*. Proceedings of Infocom 2002, New York, 2002.

[12] Y. Ofek, *Generating a Fault Tolerant Global clock using High Speed Control Signals for the MetaNet Architecture*. IEEE Transactions on Communications, 42(5), pp2179-88, 1994.

[13] L. Hu, *Distributed Code Assignments for CDMA packet radio networks*. IEEE ACM Transactions on Networking, pp. 668-677, Dec 1993.

[14] J.J. Garcia-Luna-Aceves and J. Raju, *Distributed Assignment of Codes for multi-hop Packet Radio Networks*. Proceedings of MILCOM 1997, Monterey, California 1997.

[15] T. Salonidis, P. Bhagwat, L. Tassiulas, R.O. LaMaire, *Distributed Topology Construction of Bluetooth Personal Area Networks*. Proceedings of Infocom 2001, Anchorage Alaska, 2001.

[16] C. Law, A. K. Mehta, and K. Siu, *Performance of a new Bluetooth scatternet formation protocol*. Proceedings of IEEE/ACM MobiHoc, Long Beach CA, Oct. 2001.

[17] G.V. Zaruba, S. Basagni, I. Chlamtac, *Bluetrees - scatternet formation to enable Bluetooth-based ad hoc networks*. IEEE International Conference on Communications (ICC) 2001, pp. 273-277.

[18] D. Baker and A. Ephremides, *The architectural organization of a packet radio network via a distributed algorithm*. IEEE Transactions on Communications, COM-29 (1981), pp. 1694-1701.

[19] M. Gerla and J. T.-C Tsai, *Multicluster, mobile multimedia radio network*. ACM Baltzer J. Wireless networks, vol. 1, no. 3, pp. 255-265, 1995.

[20] Bluetooth Special Interest Group,*Specification of the Bluetooth system, ver 1.0B*. www.bluetooth.com, October 2000.

[21] Bluetooth Special Interest Group,*Specification of the Bluetooth system, version 1.2*. www.bluetooth.com, 2003.

[22] V. Bharghavan, S. Shenker, L. Zhang, *MACAW: A media Access protocol for wireless LANs*. Proc. ACM Sigcomm 94.

[23] IEEE, *Wireless LAN, Medium Access Control (MAC) and Physical Layer (PHY) specifications*. IEEE Standard, 1999.

[24] A. Charny, *An algorithm for Rate Allocation in a packet Switching network with feedback*. MS Thesis, MIT May 1994.

[25] L. Kalamboukas, *Congestion Management in High Speed Networks*. PhD Thesis, University of California Santa Cruz, September 1997.

[26] A. Demers, S. Keshav, and S. Shenker, *Analysis and Simulation of a Fair-queueing Algorithm*. Proc. ACM SigComm 89, pp 1-12, Vol. 1, No. 1, 1989.

[27] J. Rexford, F. Bonomi, A. Greenberg, and A. Wong, *Scalable architecture for integrated traffic shaping and link scheduling in high-speed ATM switches*. IEEE Journal on Selected Areas in Communications, Vol. 15, No. 5, June 1997, pp. 938-950.

[28] V. Park, M. S. Corson, *A performance comparison of the Temporally-Ordered Routing Algorithm and Ideal Link State Routing*. International Symposium on Computer Communications (ISCC) '98, Athens, Greece 1998.

[29] D. Bertsekas, R. Gallager, *Data networks*. Prentice Hall, London, U.K., 1992.

[30] ——, *Distributed on-line schedule adaptation for balanced slot allocation in Bluetooth scatternets and other ad hoc network architectures.* Technical report.

[31] Y. Xue, B. Li, K. Nahrstedt, *Price-based Resource Allocation in Wireless Ad Hoc Networks.* Proc. 11th International Workshop on Quality of Service (IWQoS) 2003, Monterey, CA, USA.

[32] S. Sarkar and L. Tassiulas, *End-to-end bandwidth guarantees through fair local spectrum share in wireless ad-hoc networks.* Proc. Control and Decision Conference (CDC) 2003, Maui, HI, USA.

[33] A. Kumar and R. Gupta, *Capacity Evaluation of Frequency Hopping Based Ad-hoc Systems.* Proc. SIGMETRICS, 2001.

APPENDIX A: Proof of Convergence Theorem

We assume that every link in the network will be asynchronously activated for rate adjustment infinitely often. This means that the links do not stop attempting to perform rate adjustments and intervals in-between consecutive rate adjustments of a specific link are finite. For simplicity, we assume that all links are backlogged. The proof for the case where constrained links exist follows a similar reasoning.

Let the link rate adjustment process start at time t_0. Consider the set of most constrained nodes $N^{(0)}$, for which the ratio $C_k/|L(k)|$ is equal and minimum:

$$N^{(0)} = \{i : i = arg \min_{k \in N} \{C_k/|L(k)|\}\}.$$

When a link l of node i in $N^{(0)}$ is activated for rate adjustment:

- Node i is always the bottleneck node for l because it offers the minimum deficit.
- According to the FDC algorithm of i, link l will belong to the maximum rate set of the new rate allocation $\boldsymbol{r}_i$. Also, the cardinality of the new maximum rate set of node i increases by one link.

When all adjacent links of i have been activated for rate adjustment, its maximum rate set will have $|L(i)|$ links, each link allocated a rate of $C_i/|L(i)|$. From that point on, when a link $l \in L(i)$ is activated for rate adjustment, i will be giving it a fairness deficit of zero, and no further rate adjustment will take place for such a link. Since links are activated infinitely often for rate adjustment, there will be a point $t_1 > t_0$ where all adjacent links to all nodes i in $N^{(0)}$ have been allocated a rate of $C_i/|L(i)|$.

Let $L^{(0)}$ be the set of all links adjacent to the nodes in $N^{(0)}$ and consider the algorithm operation after time t_1.

Nodes in $N^{(0)}$ will never adjust the rates of their adjacent links. When a node in $N - N^{(0)}$ executes the FDC algorithm for an adjacent link l not in $L^{(0)}$, it may decrease the rates of other adjacent links except those in $L^{(0)}$–these links have the global minimum rate in the network and will never belong to the maximum rate set during the FDC computation. This is equivalent to saying that links in $L^{(0)}$ and the bandwidth they consume have been ”removed” from the network; the nodes in $N - N^{(0)}$ redistribute their remaining capacity to their adjacent, non-saturated links.

After time t_1, denote by $N^{(1)}$ the set of the next most constrained nodes in the network:

$$N^{(1)} = \{i : i = arg \min_{k \in N - N^{(0)}} \{(C_k - \sum_{j \in N^{(0)}} r_{kj})/|L(k)|\}\}.$$

When a link $l = (i, j)$ adjacent to a node $i \in N^{(1)}$ is activated for a rate adjustment:

- If the other endpoint node j is in $N^{(0)}$, no rate reallocation takes place because the link fairness deficit is zero.
- Otherwise, node i is the bottleneck node for this link. Now if there is another link in $L(i)$ for which the endpoint node $k \neq j$ is in $N^{(0)}$, then its rate cannot be decreased further by the FDC algorithm of i because it has already established the minimum possible fair share in the network $(C_k/|L(k)|)$.
- The cardinality of the new maximum rate set of node i increases by one link.

Now let time instant $t_2 > t_1$ be the point in time where all adjacent links to all nodes i in $N^{(1)}$ (except the links (i, k) for which node k is in $N^{(0)}$) will have been allocated a rate of $C_i/|L(i)|$. We can easily show by induction that there exists a future finite time instant t_{n+1} until every set of constrained nodes

$$N^{(n)} = \{i : i = arg \min_{k \in N - N^{(0)} \cup \ldots \cup N^{(n)}} \{(C_k - \sum_{j \in N^{(0)} \cup \ldots \cup N^{(n)}} r_{kj})/|L(k)|\}$$

will saturate its remaining links. It follows that the algorithm converges to the MMF allocation in a finite number of steps.

APPENDIX B: Algorithm pseudocodes

```
Procedure ComputeFairnessDeficit
Input: i , $r_i$, j, $B_{ij}$      Output: $r'_i$ , $fd_{i->j}$
Initialization: $t=0$, $r'_i = r_i$ , $E_i=C_i-Sum_j\{r_{ij}\}$
begin
1. $r'_{ij} = r'_{ij} + E_i$ /*Increase by the available node bandwidth*/
2. max_rate = $\max_{\{k \text{ in } N(i)\}} r'_{ik}$
3. while ($r'_{ij}$ < max_rate AND $r'_{ij} < B_{ij}$ )
   begin
    3.1.  $t = t + 1$
    3.2.  max_rate = $\max_{\{k \text{ in } N(i)-\{j\}\}} r'_{ik}$
    3.3.  $M^{(t)} = \{(i,j_1), ..., (i,j_m): r'_{ij1} = ... = r'_{ijm} = max\_rate\}, m=| M^{(t)}|$
    3.4.  $r'_{ij1} = ... = r'_{ijm} = r'_{ij} = ( r'_{ij1} + ... + r'_{ijm} + r'_{ij}) / (m+1)$
    end
4. If ($r'_{ij} > B_{ij}$) /*rate constraint is less than the fair share */
    4.1. $r'_{ij} = B_{ij}$
    4.2. $r'_{ijk} = r'_{ijk} + (r'_{ij} - B_{ij}) / m$ for every $(i, j_k)$ in $M^{(t)}$.
5. $fd_{i->j} = r'_{ij} - r_{ij}$
end
```

(a)

```
Procedure AssignSlots
Input: $i$ , $\mathbf{x}_i$, $\mathbf{S}_i$ , $j$, $\mathbf{S}_j$ , $T$      Output: $\mathbf{S}_i$ , $\mathbf{d}_i$
Initialization: $\mathbf{S}_i = \mathbf{S}_i$ , $\mathbf{d}_i$ (s)= 0 for s = 0, &, T -1
begin /*Phase I: Match the idle slots of the other end j and assign  on link (i,j)*/
1. Slot position set $I_0 = \{s: \mathbf{S}_i (s) = idle$ AND $\mathbf{S}_j (s) =$ idle, s=0,&,T -1$\}$
 1.1.repeat /*First match the slots that are idle in both $\mathbf{S}_i$ and $\mathbf{S}_j$*/
      Randomly select a slot position $s$ from $I_0$
      $\mathbf{S}_i (s)=j$, $\mathbf{d}_i (s)=1$  /*Assign slot position s to link (i,j) in $\mathbf{S}_i$ */
      $x_{ij} = x_{ij}$   1,  $I_0 = I_0 - \{s\}$
     until ($x_{ij}$ == 0   OR   $I_0$ is empty)
 1.2. If ( $x_{ij}$ == 0 ) stop and exit procedure.
2. Form set of surplus links $X_i^{--} = \{(i,k): x_{ik} < 0\}$ from $\mathbf{x}_i$.
 2.1. for every link $(i,k)$ in $X_i$   begin
       Slot position set $I_k = \{s: S_i (s) = k$ AND $S_j (s) =$ idle, s=0,&,T -1$\}$
       repeat /*Match idle slots in $\mathbf{S}_j$ and ones of surplus link (i,k) in $\mathbf{S}_i$ */
        Randomly select a slot position $s$ from $I_k$
        $\mathbf{S}_i (s)=j$, $\mathbf{d}_i (s)=1$  /*Assign slot position s to link (i,j) in $\mathbf{S}_i$ */
        $x_{ij} = x_{ij}$   1, $x_{ik} = x_{ik} + 1$,  $I_k = I_k - \{s\}$
       until ($x_{ik}$ == 0   OR   $I_k$ is empty)
       If ( $x_{ij}$ == 0 ) stop and exit procedure.
     end/*end for loop 2.1.*/
                              /*Phase II starts here*/
3. for every link $(i,k)$ in $X_i$   begin
     if ($x_{ik}$ < 0) begin /*If this link has still slots to give after Phase I*/
       Form set $I_k$ by randomly selecting $x_{ik}$ slot positions $s$ : $\mathbf{S}_i (s) = k$
     for every slot position $s$ in $I_k$
       $S_i (s) = j$ , $d_i (s) = 1$ /*Assign slot $s$ to link $(i,j)$ in $\mathbf{S}_i$ */
   end /*end for loop 3.*/
end /*Procedure AssignSlots*/
```

(b)

Fig. 14. (a) The Fairness Deficit Computation (FDC) algorithm (b) The slot assignment algorithm