TECHNICAL RESEARCH REPORT

Distributed On-Line Schedule Adaptation for Balanced Slot Allocation in Bluetooth Scatternets and other Wireless Ad-Hoc Network Architectures

by Leandros Tassiulas and Theodoros Salonidis

CSHCN TR 2002-14
(ISR TR 2002-24)

The Center for Satellite and Hybrid Communication Networks is a NASA-sponsored Commercial Space Center also supported by the Department of Defense (DOD), industry, the State of Maryland, the University of Maryland and the Institute for Systems Research. This document is a technical report in the CSHCN series originating at the University of Maryland.

Web site http://www.isr.umd.edu/CSHCN/

Distributed on-line schedule adaptation for balanced slot allocation in Bluetooth scatternets and other wireless ad hoc network architectures

Theodoros Salonidis and Leandros Tassiulas
ECE department, University of Maryland at College Park
{thsalon,leandros}@eng.umd.edu

***Abstract*- In this paper we propose an algorithm for design and on the fly modification of the schedule of an ad-hoc wireless network in order to provide 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 the transmissions at each slot are explicitly specified over a time period of length *T*.**

First we 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 max-min 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 time varying topologies we show that the latter algorithm achieves balanced rate allocation in the actual slotted system that are very close to the max-min fair rates. The experiments show also that the algorithm is very robust on topology variations, with very good tracking properties of the max-min fair rate allocation.

1. 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 users. A major obstacle in this quest is the spatial contention of flows 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 flows 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 ([20], [21]) to address them. Despite their distributed nature and flexibility, random access MAC protocols cannot offer a way of strict bandwidth allocation and guarantees.

Multi-channel wireless technologies address spatial contention of flows at the physical layer where each channel is defined by a separate code or frequency. The idea is that if the flows in a vicinity do not use the same channel, then conflict-free transmissions can take place at the same time. Even if this method eliminates[1] collisions due to unintended broadcast transmissions, contention of flows still exists because each wireless node is usually equipped with a single transceiver and cannot simultaneously transmit or receive in more than one link flow. This form of contention necessitates coordination of the node transmissions on channels and flows by establishing conflict-free link schedules [1]. According to such a schedule two link flows that share the same wireless node are not allowed to transmit simultaneously. Also, nodes must be synchronized to communicate on common flows at the same time. 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 flow determines the rate (bandwidth) allocated to this flow.

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.

[1] Interference is never totally eliminated but tolerated up to a certain degree that depends on the physical layer implementation of the reference technology and the locality of transmissions. We assume elimination of this kind of interference i.e. no channel errors during a conflict-free transmission.

The emergence of the Bluetooth multi-channel technology [19] 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] trade-off perfectly conflict-free transmissions for lower complexity and better robustness in dynamic network operation. The downside here is that this results to a lack of ability to provide bandwidth allocation guarantees.

In this paper we introduce a low complexity "hard cordination" distributed algorithm that aims in establishing and maintaining maxmin fair bandwidth allocations in any slotted multi channel wireless network, including Bluetooth scatternets. 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 flows. If a flow cannot use all the bandwidth because of a constraint, then the residual bandwidth is distributed to less constrained ones. Among any feasible bandwidth allocations, a maxmin fair one ensures that the most constrained flows are allotted the maximum possible bandwidth.

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 considered 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 convergence. Through extensive simulations in dynamic networks we show that the algorithm possesses very good tracking properties of the max-min fair rate allocation.

The rest of the paper is organized as follows. Section 2 presents the network model and definition of max-min fairness. Section 3 introduces the fluid part of the asynchronous algorithm that computes the amount of rate adjustments. Section 4 describes the scheduling technique that enforces these rate adjustments by means of conflict-free slot reallocations. Section 5 provides experiments where the algorithm performance is evaluated. Section 6 provides algorithm extensions, section 7 discusses related work and finally, section 8 concludes the paper.

2. NETWORK MODEL AND MAX-MIN FAIRNESS DEFINITION

2.1. Network and communication model

The wireless ad hoc network is represented by a graph G(N,L) and a set of logical link flows F. Each node is assumed to have a unique id (for example the node's MAC address). An edge signifies that nodes i and j are within wireless range and they have established a physical wireless link. Each physical link is associated with a number of bi-idirectional **logical link flows** of the set F. In this paper we assume that there are no end-to-end flows spanning more than one physical links in the network.

All nodes in the system are synchronized on a slot basis. Synchronization can be achieved by using GPS clocks or signaling techniques similar to those employed in wired networks [11] modified for the wireless ad hoc network setting[2]. Each system slot supports bi-directional transfer of data or control packets by means of a pair of equal duration half-duplex mini-slots.

The problem of flow contention due to unintended broadcast transmissions is avoided by means of a

[2] Certain slotted systems such as Bluetooth may not support such a synchronization mechanism. As will be evident and explained later, the algorithm does not rely on system-wide synchronization for correct operation.

distributed channel code assignment scheme running at the physical layer [13][12]. In Bluetooth this function is provided by scatternet formation protocols [14][15][16]. Still, contention of flows exists because each wireless node cannot simultaneously communicate in more than one physical link. To implement conflict-free communications, each node *n* maintains a local link schedule $\mathbf{S}_n$ of period *T*. In every slot of $\mathbf{S}_n$, node *n* can either communicate on a single flow or remain idle. Transmission on a flow *f* is conflict-free, only if both ends agree to communicate on *f* on the same slots of their local link schedules.

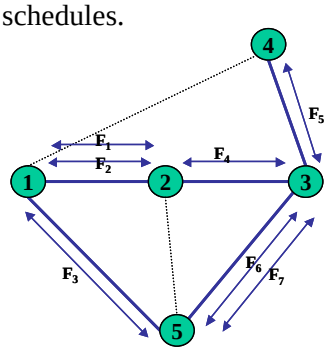


Figure 1: Dotted lines denote wireless proximity but no established physical link. Flows F1 and F5 can transmit simultaneously without conflict even if nodes 1 and 4 are within transmission range. Still every node can transmit to or receive from only one flow at a time. For example, flows F4 and F5 sharing node 3 and Flows F1 and F2 sharing physical link (1,2) cannot transmit simultaneously.

We use two models to represent bandwidth allocation. In the **slot model** the bandwidth allocated to a flow *f* is expressed as the number of slots τ_f in a *T*-slot periodic conflict-free link schedule. The **fluid model** does not refer to a slotted system. The bandwidth r_f allocated to a flow *f* is the long-term fraction of time a link spends communicating conflict-free on this flow. The two models serve different purposes. The fluid model is more general and intuitive and can be used to describe notions such as feasibility and max-min fairness. However a real system will always work in the discrete domain on a finite *T*-periodic schedule.

2.2. Feasibility and maxmin fairness definition

Under the fluid model, the **effective capacity** C_n of a node *n*, is defined as the maximum bandwidth that a node provides its flows for communication. If C_n is less than unity, then the node is always partially utilized by the flows sharing it and remains idle for the rest of the time.

A bandwidth allocation of flows $R=(r_1,\ldots,r_f,\ldots,r_n)$ is called **feasible** if there exists a conflict-free (not necessarily periodic) schedule that allocates to every flow *f*, a long-term rate equal to r_f, $1\le f\le n$. The set of all feasible bandwidth allocation vectors defines the feasibility region, which can be characterized by a set of constraints. Since a node cannot communicate on different flows simultaneously, it is obvious that one constraint would be that the sum of the rates of all flows sharing a node 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, [1] proves that a node capacity equal to 2/3 provides with a sufficient (albeit not necessary) characterization of feasibility. We therefore reach the following node **capacity constraint** for feasibility:

$$\sum_{f\ adjacent\ to\ n} r_f \le C_n, \quad \forall n\in N, \text{where}$$

$$C_n=\begin{cases}1, & if\ network\ graph\ G\ is\ bipartite\\ 2/3, & otherwise\end{cases} \qquad (1)$$

If a flow *f* has a long-term arrival rate ρ_f then we also need a **demand constraint** on the maximum allowable rate for this flow:

$$r_f \le \rho_f \qquad (2)$$

A feasible rate allocation is said to be **maxmin fair** (*MMF*) if the rate allocated to a flow cannot be increased without hurting other flows having equal or less rate. In Figure 1 the MMF allocation of flows is $(r_1,r_2,r_3,r_4,r_5,r_6,r_7)=(1/3,1/3,1/3,1/4,1/4,1/4,1/4)$. We see that because node 3 is fully utilized, the rate of ¼ allocated to flow 4 cannot be increased without hurting the rates of the flows 5,6 and 7 that share node 3 and have been assigned an equal rate.

More formally a rate allocation vector **r** is defined to be maxmin fair if:

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

Given a feasible rate allocation on a network it is very useful to have a distributed criterion to test if this

allocation is maxmin fair or not. A node *n* is defined as a **bottleneck node** of flow *f* if the sum of the rates of all flows equals the node effective capacity C_n and the rate of flow *f* is greater than or equal to the rate of all other link flows of node *n*.

MMF criterion: A bandwidth allocation is maxmin fair (MMF) if and only if every flow *f* satisfies at least one of the following conditions:

- The bandwidth allocated to the flow equals its long-term arrival rate ρ_f.
- The flow f has at least one bottleneck node.

For example in Figure 1 we can easily verify that nodes 1 and 3 are the bottleneck nodes for the flow sets {F1,F2,F3} and {F4,F5,F6,F7} respectively.

3. A DISTRIBUTED ALGORITHM FOR THE FLUID MODEL

3.1. 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 maxmin fair solution. For clarity we will consider the simplest version of the algorithm for the case when there is only one logical flow per physical link in the network. The extensions of multiple logical flows per physical link are discussed in section 6. In this case the rate r_f of logical flow *f* on physical link *(i,j)* is denoted as r_{ij}. If $N(i)$ is the set of one-hop neighbors[3] of node *i*, then the link rate allocation $\mathbf{r}_i$ for node *i* is defined as the set $\mathbf{r}_i = \{r_{ij} > 0, \quad \forall j \in N(i)\}$. The **capacity feasibility constraint** for node *i* can then be expressed as:

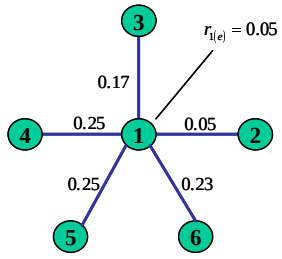
$$\sum_{j \in N(i)} r_{ij} \le C_i . \qquad (3)$$

The **available node bandwidth** according to link rate allocation $\mathbf{r}_i$ is then defined as:

$$r_{i(e)} = C_i - \sum_{j \in N(i)} r_{ij} \qquad (4)$$

Initially, the algorithm starts from an arbitrary feasible rate allocation $\boldsymbol{R}$ in the network. At asynchronous points in time a link flow is activated[4] for a possible rate adjustment. The adjustment is such that at least one of the link endpoints becomes a bottleneck node for the link. A bottleneck node can be created if the link rate 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 **link fairness deficit**.

Starting from $\mathbf{r}_i$, the **fairness deficit computation (FDC)** algorithm for link *(i,j)* works iteratively and finds a new allocation $\mathbf{r}_i'$ such that eventually the rate r_{ij}' belongs to the maximum link rate set of $\mathbf{r}_i'$. Then fd_{ij} is by definition equal to $r_{ij}' - r_{ij}$. Figure 2 is a representative example of the algorithm operation, while Figure 3 (see Appendix) contains the detailed algorithm pseudocode, which includes the case when there is a upper demand constraint ρ_{ij} on the flow of link *(i,j)*.



step#	r_{12}'	r_{13}'	r_{14}'	r_{15}'	r_{16}'	$r_{1(e)}'$	max_rate
0	0.05	0.17	0.25	0.25	0.23	0.05	0.25
1	0.1	0.17	0.25	0.25	0.23	0	0.25
2	0.20	0.17	0.20	0.20	0.23	0	0.23
3	0.215	0.17	0.20	0.20	0.215	0	0.215

Figure 2: The FDC algorithm for link *(1,2)* by node 1. At each step we consider the maximum rate set *M* (denoted by the shaded entries). If r'_{12} does not belong in *M*, the total bandwidth of the links in *M* and link *(1,2)* is equally distributed to them. This process continues until link *(1,2)* is in *M*. The last row is the new rate allocation $\mathbf{r}_1'$ and the fairness deficit is $fd_{1\to 2} = 0.215 - 0.05 = 0.165$.

3.2. 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 $fd_{i\to j}$ and $fd_{j\to i}$ on link *(i,j)* and exchange their deficit values. The **link fairness deficit** is defined as $fd_{ij} = \min(fd_{i\to j}, fd_{j\to i})$.

[3] By "one-hop neighbors" we refer to the nodes in range of node *i* for which a physical link has been established.

[4] We use the term "link activation" with respect to the rate adjustment process. The link is always active to be used for communication according to the local link schedules of its endpoints.

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 $\mathbf{r}_i^{'}$ is the one given by the FDC algorithm of $fd_{i\to j}$ in step 1. The maximum deficit node *j* reaches its new link rate allocation $\mathbf{r}_j^{'}$ 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}.

Theorem 1: Given a static topology and an arbitrary initial feasible network rate allocation ***R***, the above algrotihm converges to the network max-min fair solution after a finite number of link activations for rate adjustment.

Proof: see Appendix.

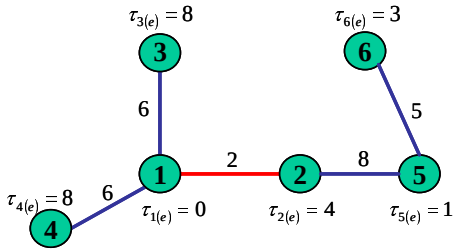
The algorithm termination is totally distributed and no explicit message needs to be sent to the entire network to signal convergence: when a link is activated for possible rate adjustment, its rate is adjusted only if the link fairness deficit is non-zero.

4. THE ALGORITHM FOR THE SLOTTED SYSTEM

4.1. Fairness deficit computation and slot assignment algorithm

The fluid algorithm guarantees convergence to the network max-min 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.

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 schedule *S* 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.



$T = 14$

slot# / id	0	1	2	3	4	5	6	7	8	9	10	11	12	13
S1	**4**	3	3	4	3	4	3	4	2	3	2	4	3	4
S2	-	5	5	5	5	5	5	5	1	5	1	-	-	-
S3	-	1	1	-	1	-	1	-	-	1	-	-	1	-
S4	**1**	-	-	1	-	1	-	1	-	-	-	1	-	1
S5	**6**	2	2	2	2	2	2	2	6	2	-	6	6	6
S6	**5**	-	-	-	-	-	-	-	5	-	-	5	5	5

Figure 4 and Table 4: A wireless ad hoc network using the T=14 periodic conflict-free schedule of Table 4. Each slot entry *j* in the local schedule $\mathbf{S}_i$ means communication of node *i* on link *(i,j)*.

The slotted fairness deficit computation algorithm for node *i*, uses the one of the fluid model to reach from discrete slot allocation τ_i to $\tau_i^{'}$, and outputs the rate difference vector $\mathbf{x}_i = \tau_i^{'} - \tau_i$. An example of the detailed operation of the slotted FDC is shown below:

step		(1,2)	(1,3)	(1,4)	rem	Actions
0	τ_1	2	6	6	0	T=14
1	$\mathbf{r}_1$	2/14	6/14	6/14	0.0000	$r_{1j} = \tau_{1j} / T$
2	$\mathbf{r}_1^{'}$	0.3333	0.3333	0.3333	0.0000	FDC algorithm
3	$\tau_1^{'}$	4	4	4	2	$\tau_{1j}^{'} = \lfloor r_{1j}^{'} \cdot T \rfloor$
4	$\tau_1^{'}$	6	4	4	0	Randomly distribute remainder slots to last maximum rate set
5	$\mathbf{x}_1$	+4	-2	-2	0	$x_{1j} = \tau_{1j}^{'} - \tau_{1j}$

Table 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 $\mathbf{x}_1$. The fairness deficit is 4 slots.

Given $\mathbf{x}_i$, 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 current τ_{ik} slot

positions will be 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 $\boldsymbol{S}_j$ and $\boldsymbol{S}_i$, 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.

Procedure ***AssignSlots(*** i , $\mathbf{x}_i$, $\mathbf{S}_i$, j, $\mathbf{S}_j$, T, , $\mathbf{S}_i$' , $\mathbf{d}_i$**)**
Input: i , $\mathbf{x}_i$, $\mathbf{S}_i$, j, $\mathbf{S}_j$, T **Output:** $\mathbf{S}_i$', $\mathbf{d}_i$
Initialization: $\mathbf{S}_i' = \mathbf{S}_i$, $\mathbf{d}_i = \mathbf{0}$
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, $0 \le s \le T-1\}$
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**)** */*end for loop 2.3.*/*
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, $0 \le s \le T-1\}$
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*/*

Figure 5: The slot assignment algorithm

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 the new link schedule of *i*, and a list indicating the (extra) slot positions that should be assigned to link *(i,j)*.

In Table 6 node 1 is called to decide on the extra slot positions that will be assigned to link *(1,2)*.

Slot#	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$\mathbf{S}_1$	4	3	3	4	3	4	3	3	2	3	2	4	3	4
$\mathbf{S}_2$	-	5	5	5	5	5	5	-	1	5	1	-	-	-

Table 6: Idle slot positions {7,12} 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} .

The rate difference vector (row 5 in Table 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)*.

4.2. Signaling schedule updates

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

- A field specifying if the packet is an "increase" (SC_inc) or "decrease" (SC_dec) SC packet.
- A list of slot positions that need to be modified in the receiver's local schedule (Represented by a *T*-bit vector). For an SC_inc packet the indicated positions will be assigned to link *(i,j)* in the receiver *j's* updated schedule, while for an SC_dec packet they will be assigned as idle.
- The number of slots the receiver should wait before applying the above schedule update.

Starting from slot *s* where the link was activated for rate adjustment, the **commit slot offset** $coff_{ij}^{(s)}$ 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 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.

4.3. 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 $a = b_i^{(s)}(\{j\})$ slots and according to its own schedule $\mathbf{S}_j$, it needs $b_j^{(s+a)}(N(j) - \{i\})$ additional slots to update the rest of its neighbors. Therefore starting from slot *s*, node *j* will need a total of $B_j^{(s)} = a + b_j^{(s+a)}(N(j) - \{i\})$ slots to propagate the schedule update. The commit slot offset is the number of slots until both *i* and *j* reach all their neighbors: $coff_{ij}^{(s)} = \max(A_i^{(s)}, B_j^{(s)})$.

Referring to Table 6, assume that node 1 has just performed the slot assignment algorithm at slot s=8. Given $\mathbf{S}_1$, 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 $\mathbf{S}_2$ it will need $b_2^{(10)}(\{5\})$=5 additional slots to reach node 5 (on slot 1 of its periodic schedule). Thus, starting from slot s=8 node 2 will need $B_2^{(8)} = 2 + 5 = 7$ slots for the schedule update propagation and finally the commit slot offset is $coff_{12}^{(8)} = \max(A_1^{(8)}, B_2^{(8)}) = 7$ slots.

4.4.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 control packets (termed as FD packets). The FD packet sent by each node *x* contains the following information:
   - The node's calculated discrete fairness deficit with respect to link *(i,j)*.
   - The number of slots $B_x^{(s)}$ node *x* needs to propagate the schedule update to all its neighbors in case it turns out to be the maximum deficit node.
   - A *T*-bit vector $\mathbf{I}_x$ indicating the idle slot positions in its own link schedule $\mathbf{S}_x$.
2. If any of the two fairness deficits is zero, then 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.

If *i* is the minimum deficit node, then based on the FD packet received by *j*:

- Given $\mathbf{I}_j$, 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 $coff_{ij}^{(s)} = \max(A_i^{(s)}, B_j^{(s)})$.

4. 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 schedule 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*.
5. 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 and the schedule adjustment is complete.

Slot 8: (1->2): FD[$fd_{1->2}$ = 2, [B_1 = 4], $\boldsymbol{S}_1$]
(2->1): FD[$fd_{2->1}$ = 5, [B_2 = 7], $\boldsymbol{S}_2$]

- $fd_{1->2} < fd_{2->1}$: Node 1 is the minimum deficit node.
  Node 1 executes the slot assignment algorithm with inputs $\mathbf{x}_1$ (4th row of Table 4), $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ (Table 5) and decides on slot positions {7,11,12,13} to be assigned to link *(1,2)* in $\boldsymbol{S}_1$.
  Node 1 computes A_1 =3 and $coff_{12}^{(8)}$ = max(A_1, B_2) = 7 and sets a commit timer to expire after 7 slots.

Slot 9: (1->3): SC[*decrease_rate*, slot_pos_list = {7,12}, commit_*after* = 6 slots]
Node 3 sets a commit timer to expire after 6 slots.
Slot 10: (1->2): SC[*increase_rate*, slot_pos_list = {7,11,12,13}, commit_*after* = 5 slots]
Node 2 sets a commit timer to expire after 5 slots.
Slot 11: (1->4): SC[*decrease_rate*, slot_pos_list = {11,13}, commit_*after* = 4 slots]
Node 4 sets a commit timer to expire after 4 slots.
Slot 15: (2->5): SC[*decrease_rate*, slot_pos_list = {NIL}, commit_*after* = 0 slots]

All commit timers expire, and nodes 1,2,3,4,5 update their local link schedules.

Figure 6: Actions performed after activation of link (1,2) at slot 8 of the schedule in Table 4.

Figure 6 illustrates the system evolution after the activation of link *(1,2)* during slot 8 of the schedule in Table 4.

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 S. If multiple links happen to be activated for a rate increase at the same slot, the slot reassignment is conflict-free because the additional slots are given to links that do not have common node endpoints. This follows from the conflict-free property of the current schedule that activates links that constitute a matching in the network topology graph. Also a node applies the following rules for updating its local schedule during the schedule adjustment process:

1. When a node *j* receives an SC_inc packet from *i*, it modifies its local schedule by unconditionally assigning to link *(i,j) all* the slot positions seen in the packet list.
2. When a node *j* receives an SC_dec packet from *i*, it reassigns to idle only the positions that are currently assigned to link *(i,j)* and ignores the rest.

If node *j* updated unconditionally in rule (2) then, if it received an SC_inc packet from *i* and then an SC_dec packet from *k*, and also these packets happened to have common positions to be modified in *j*'s schedule, then *j* would first give these slots to link *(i,j)* and then set them idle. Then *i* would not know about the second change in j's schedule and there would be a conflict on these slots between them. An example of this situation can be seen in Figure 4, for the rate adjustment of links *(1,2)* and *(5,6)* where node 2 may receive an SC_inc from 1, then an SC_dec from 2 and both packets contain slot position 7 to be modified in 2's local schedule. By applying the above rule, node 2 will ignore slot position 7 in the SC_dec packet, yielding the proper assignment. It is easy to verify that the rule maintains the conflict-free property regardless of the ordering and type of control packets and the indicated slot positions in them.

4.5. Protocol communication Requirements

The amount of control information needed by the protocol depends only in the system period *T* and not on 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:

$$B_{control} = 2\lceil \log_2 T \rceil + T \text{ bits} \qquad (5)$$

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

5. PERFORMANCE EVALUATION

5.1. Experimental setting

Topology dynamics are modeled by having links going up and down in a static baseline topology [26]. This simplified 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:

1. **Relative computation error**: If the MMF rate of a link *(i,j)* at time *t* is $r_{ij}^m(t)$ and the computed rate is $r_{ij}(t)$, the relative computation error for link *(i,j)* is $\left|1 - r_{ij}(t) / r_{ij}^m(t)\right|$ at time *t*. For each slot *t*, we consider the **average** relative computation error over all currently ACTIVE links. After each topology change, the reference MMF rates $r_{ij}^m(t)$ are computed off-line using a centralized algorithm similar to the one of [27] for wireline networks appropriately adapted to our wireless setting. The detailed algorithm can be found in Appendix C.

2. **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 rate feasibility region is defined exactly in this case. Such topologies arise in clustered architectures [17] [18] [19] where each cluster (channel) is defined and controlled by a master node and the rest of the cluster members are slaves. Inter-cluster communication is performed by slave gateway nodes that participate in more than one clusters. In the non-bipartite case nodes can set their effective capacity to 2/3 to guarantee feasible rate allocations and the algorithm will still yield MMF rates but with respect to this fractional capacity.

We used an N=100 node bipartite baseline topology with 50 nodes per set. This yields 2500 links in the baseline topology. As system technology parameters we use 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 period of is T=200 slots, which is the maximum that could be supported by the current Bluetooth specification[5]. In terms of traffic demands, all link flows are assumed backlogged (no demand constraints) when ACTIVE.

5.2. Experiments in Static Networks

All simulations in static topologies were run for 100000 slots. In the first experiment every node has a degree of D_{max} (p=1.0) and the target MMF rate every link in the network must reach is $1/D_{max}$ (approximated by T/D_{max} slots in each case).

Figure 7 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.

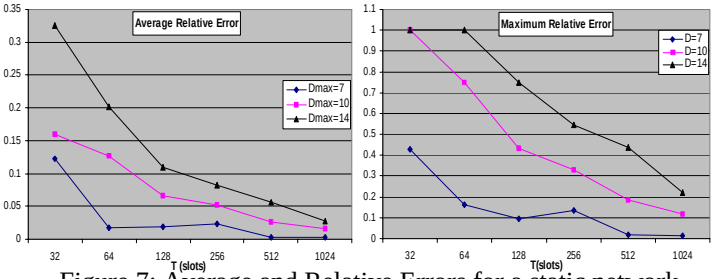


Figure 7: Average and Relative Errors for a static network of N=100 p=*1.0* and T_{adjust}=512 slots for various choices of T and D_{max}. The average and maximum relative errors are computed over all active links at the last slot of the simulation.

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 8 as the period T increases.

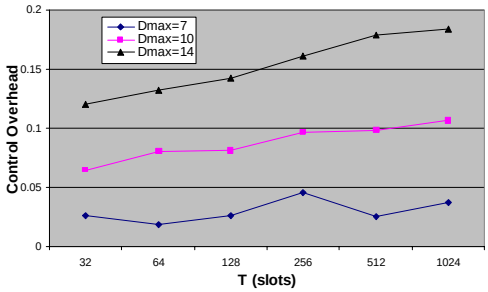


Figure 8: Control Overhead for a static network of N=100, p = 1.0 and T_{adjust}=512 slots for various choices of T and D_{max}.

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 7 and 8. 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 9 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

[5] 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 (5), we can see that the maximum period T for DH1 packets is 200 slots and for DM1 packets 122 slots.

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.

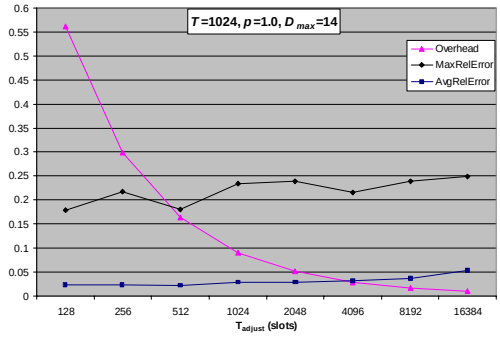


Figure 9: Effect of reduction in the frequency of link rate adjustments.

Figure 10, 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.

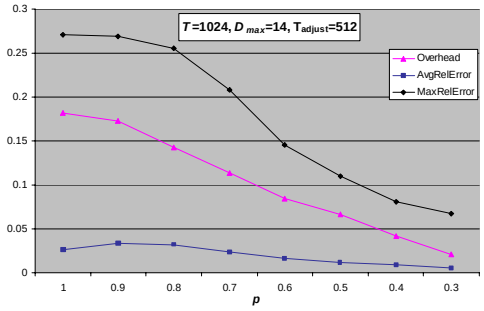


Figure 10: Effect of the density topology parameter p.

5.2. Experiments in dynamic networks

For dynamic networks, all simulations run for 500000 slots. We study the distribution (pdf) 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 Figure11a, 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 (1536000slots) to 1min (48000slots). 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%.

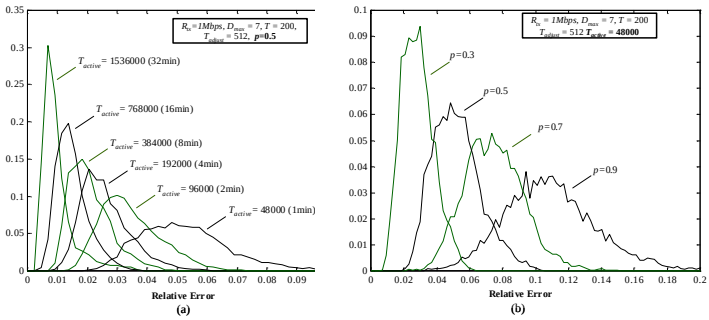


Figure 11: Effect of (a) rate of topology changes and (b) topology density in the distribution of the average relative error.

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.

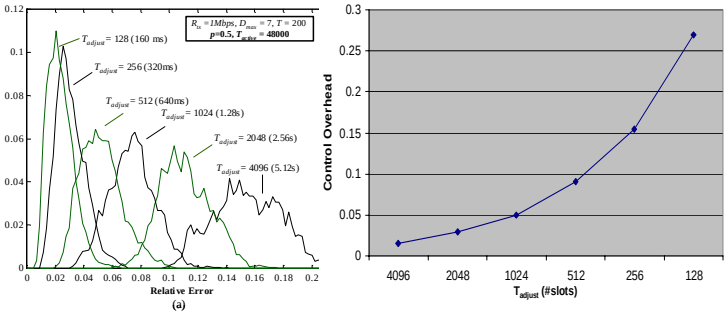


Figure 12: Effect of frequency of link activations on (a) the average relative error and (b) control overhead.

Figure 12 shows the effect of the rate adjustment parameter T_{adjust} in the most dynamic case where links

form and fail every 1minute (48000 slots) on the average. As T_{adjust} varies from 5.12s (4096slots) to 160ms (128 slots), the error mean and variance decrease slightly (Figure 12a) but the control overhead increases (Figure 12b). For T_{adjust} =160ms (128slots), 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 performance degradation of the algorithm for technologies using a certain radio transmission rate and wish to support a larger maximum number of MMF flows per node in a dynamic network.

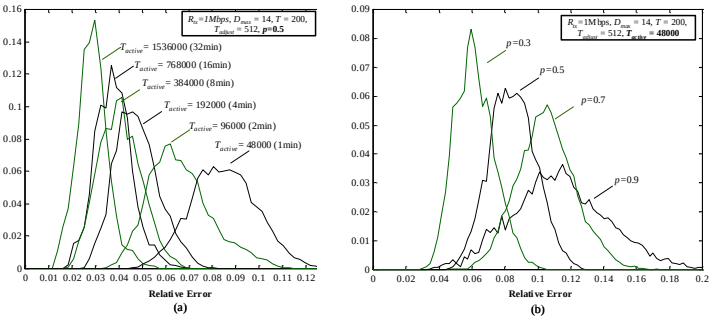


Figure 13: D_{max}=14: Effect of (a) rate of topology changes and (b) topology density in the distribution of the average relative error.

Technologies supporting higher transmission rates result in 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.

6. EXTENSIONS

6.1. Multiple logical flows per physical link

In the case of multiple logical flows per physical link, each time a link *(i,j)* is activated for rate adjustment, the link endpoint nodes compute a per-flow fairness deficit for each flow of link *(i,j)*. The deficit calculation algorithm for a flow will change only in that instead of link rates a node considers the rates of all its adjacent flows. If all flow fairness deficits for link *(i,j)* are zero, no adjustment needs to take place. If more than one flow deficits are non-zero then only one flow rate will be adjusted during this link activation (because each flow fairness deficit is computed independently of the others). Which one to adjust can be decided in a round robin fashion.

6.2. The case of Bluetooth

Throughout the paper we assumed a mechanism that keeps the system synchronized on a slot basis. Such a mechanism is not supported in Bluetooth. Each Bluetooth device has its own "native" hardware clock, but different native clocks are not necessarily synchronized. During link formation the endpoint nodes acquire the phase between their native clocks and one node assumes the role of master and the other acts as slave. Masters provide their native clock as the time reference for communication within their channel. Each slave uses the phases with respect to its masters to know the slot boundaries where communication happens in each channel.

Our protocol can operate properly even in this setting. The local schedule of a node is with respect to its native clock tick and the node uses it to know which link it can communicate conflict-free at any time (either as a master or slave). Also according to the protocol a node is informed by FD packets about other nodes' local schedules and is instructed by SC packets to modify its own. Therefore slot positions in an FD packet refer to the native clock reference of the sender node, while slot positions in an SC packet refer to the native reference of the receiver node. Each time a control packet is sent on a link, the receiver of an FD packet (or sender of an SC packet), perform the time reference translation on the slot positions in the packet using the corresponding link phase.

7. RELATED WORK

Maxmin fairness has been addressed in both single channel and multi-channel wireless networks. 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 could also reach maxmin allocation via the appropriate flow weights. However,

the weight computation requires knowledge of the maxmin fair rates. This requires a global network MMF rate pre-computation phase, something not efficient in a large dynamic network. Nandagopal et al. [6] define fairness in terms of maximizing total logarithmic user utility functions and the resulting allocation is proportionally fair. Maxmin fairness is mentioned as a limiting case of the general utility fairness model. A centralized and a distributed algorithm targeted specifically 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 bandwidth allocations 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 (only long-term fairness).

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] is a distributed scheduling algorithm for Bluetooth scatternets but it cannot apply to the maxmin fairness objective. In DSSA every node starts with an assumption of local link traffic demands and heuristically tries to reach a conflict-free schedule of short length that satisfies them. However maxmin fairness is a global objective. Hence to use this algorithm one must first pre-compute the network maxmin fair shares and then provide them as local traffic demands to every node in the network, something not practical.

Finally, distributed algorithms for maxmin fair end-to-end session rate computation have also been studied extensively in the wireline networks context [22][23]. 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 [24][25]. 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.

8. CONCLUSIONS/FUTURE WORK

Future deployment of wireless ad hoc networks calls for decentralized techniques that allocate efficiently the scarce wireless medium to users. In this paper we present an asynchronous distributed algorithm that aims for maxmin fair bandwidth allocation of flows in a wireless ad hoc network. The algorithm is of low complexity and applies to any slotted wireless ad hoc network using multiple channels at the physical layer. Bandwidth allocations are realized by perfectly conflict-free periodic link schedules. This implies that the algorithm possesses both short-term (to the extend of the period T) and long-term fairness properties.

The distributed scheduling mechanism is driven by the rate calculation algorithm, which converges to the maxmin fair solution under the fluid model. Still, when emulating the fluid algorithm in the slotted world the convergence is not exact and there are certain restrictions and trade-offs a designer has to take into account. To this end we provide a simple analysis of the algorithm communication requirements and how they affect the spectrum of design choices of a multi-channel technology wishing to support maxmin fair bandwidth allocation of flows. As a rule of thumb, higher radio transmission rates give room for larger number of slots per period and shorter slot durations. More slots per period provide a better support for larger number of MMF flows per node while smaller slot durations yield a better performance in the face of network dynamics.

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 flow 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 simultaneous flows 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.

The low algorithm communication and computation requirements make it attractive for Bluetooth, a technology not supporting system-wide slot synchronization. While the protocol can still operate properly in this case, the convergence to the MMF rates will not be as accurate as in a synchronized system because when a slave switches channel, a slot is always wasted for aligning to the time reference and local schedule of the new channel master. Another source of overhead is the fact that the time a node spends in topology discovery using the inquiry protocol is at the expense of communication slots. We are currently engaged in an implementation that takes into account these challenges and integrates them in a complete solution.

9. REFERENCES

[1] B. Hajek and G. Sasaki, "Link Scheduling in Polynomial Time", IEEE Trans. Inform. Theory, vol. 34, no 5, Sept. 1988.

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

[3] M. Post, P. Sarachik and A Kershenbaum, "A Biased Greedy Algorithm for Scheduling Multihop Radio Networks", In 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 Max-min 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", In 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 the 2001 ACM International Symposium on Mobile ad hoc networking and computing (MobiHoc), pp. 204-211.

[11] 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.

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

[13] 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.

[14] T. Salonidis, P. Bhagwat, L. Tassiulas, R.O. LaMaire, "Distributed Topology Construction of Bluetooth Personal Area Networks", Infocom 2001.

[15] C. Law, A. K. Mehta, and K. Siu, "Performance of a new Bluetooth scatternet formation protocol", Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing 2001, Long Beach, California, USA, October 2001.

[16] 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.

[17] 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.

[18] 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.

[19] Bluetooth baseband specification v. 1.1.

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

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

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

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

[24] 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, 1990.

[25] 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.

[26] V. Park, M. S. Corson, "A performance comparison of the Temporally-Ordered Routing Algorithm and Ideal Link State Routing", ISCC '98, Athens, Greece.

[27] D. Bertsekas, R. Gallager, "Data networks"

10. APPENDIX:

Appendix A: Fairness Deficit Computation Algorithm (Figure3)

Procedure *ComputeRatesAndDeficit*(i , $\mathbf{r}_i$, j, ρ_{ij} , $\mathbf{r}_i'$, $fd_{i\to j}$)

Input: i , $\mathbf{r}_i$, j, ρ_{ij} **Output:** $\mathbf{r}_i'$, $fd_{i\to j}$

Initialization: $t=0$, $\mathbf{r}_i' = \mathbf{r}_i$, $r_{i(e)} = C_i - \sum_{j\in N(i)} r_{ij}$,

$$M^{(0)} = \{\{(i,(e))\}, \quad m = 1$$

begin

1. $r_{ij}' = r_{ij} + r_{i(e)}$ **/***Increase by the available node bandwidth***/**

2. $\max\_rate = \max_{k\in N(i)} r_{ik}'$

3. while ($r_{ij}' < \max\_rate$ **AND** $r_{ij}' < \rho_{ij}$ **)**

begin

3.1. $t = t + 1$

3.2. $M^{(t)} = \left\{(i,j_1),\ldots,(i,j_m): r_{ij_1}' = \ldots = r_{ij_m}' = \max_{k\in N(i)-\{j\}} r_{ik}'\right\}$

3.3. $\max\_rate = \max_{k\in N(i)-\{j\}} r_{ik}'$, $m = \left|M^{(t)}\right|$

3.4. $r_{ij_1}' = \ldots = r_{ij_m}' = r_{ij}' = \dfrac{r_{ij_1}' + \ldots + r_{ij_m}' + r_{ij}'}{m+1}$

end

4. If ($r_{ij}' \geq \rho_{ij}$ **)** **/***rate constraint is less than the fair share***/**

4.1. $r_{ij}' = \rho_{ij}$

4.2. $r_{ij_k}' = r_{ij_k}' + \dfrac{r_{ij}' - \rho_{ij}}{m}$ for every (i,j_k) in $M^{(t)}$.

5. $fd_{i\to j} = r_{ij}' - r_{ij}$

end

Figure 3: The fairness deficit computation algorithm

Appendix B: Proof of Theorem 1

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. The proof is for backlogged single link flows. The case of constrained flows and multiple flows per physical link is similar.

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

$$F^{(0)} = \left\{i : i = \arg\min_{k\in N}\left(\frac{C_k}{N(k)}\right)\right\}.$$

When a rate of a link *(i,j)* adjacent to a node *i* in $F^{(0)}$ is activated for rate adjustment:

- Node *i* is always the bottleneck node for this link because it offers the minimum deficit.
- According to the deficit calculation algorithm of *i*, link *(i,j)* will belong to the maximum rate set of the new rate allocation $\mathbf{r}_i'$. 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 the maximum rate set will have $N(i)$ links, each link allocated a rate of $C_i / N(i)$. From that point on, when a link *(i,j)* is activated for link rate adjustment, *i* will be giving it a fairness deficit of zero, and no further rate adjustment will take place on this link. Since every link in $F^{(0)}$ will be activated infinitely often, there will be a point in time $t_1 > t_0$ where all adjacent links to all nodes *i* in $F^{(0)}$ will have been allocated a rate of $C_i / N(i)$.

After time t_1, consider the set $F^{(1)}$ of the next most constrained nodes in the network: $F^{(1)} = \left\{i : i = \arg\min_{k\in N-F^{(0)}}\left(\frac{C_k}{N(k)}\right)\right\}$.

When a rate of a link *(i,j)* adjacent to a node *i* in $F^{(1)}$ is activated for a rate adjustment:

- If node *j* is in $F^{(0)}$, no rate reallocation takes place because the fairness deficit on link *(i,j)* is zero.
- Otherwise node *i* is the bottleneck node for this link. Now if there is another link *(i,k)* for which node *k* is in $F^{(0)}$ then its rate cannot be decreased further by the deficit calculation calculation algorithm of *i* because it has already established the minimum possible fair share in the network $C_k / N(k)$.
- 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 $F^{(1)}$ (except the links *(i,k)* for which *k* is in $F^{(0)}$) will have been allocated a rate of $C_i / N(i)$. We can easily show by induction that there will be a finite time instant t_{k+1} until every set of constrained nodes $F^{(k)} = \left\{i : i = \arg\min_{k\in N-F^{(0)}\cup\ldots\cup F^{(k-1)}}\left(\frac{C_k}{N(k)}\right)\right\}$ will saturate its remaining links. Therefore the algorithm converges in a finite number of steps.

Appendix C: Centralized algorithm for calculation of the MMF rates of a static topology.

The algorithm works iteratively and at each iteration, we consider all "bottleneck" nodes, which are defined as the nodes with the smallest capacity available per flow. We share the remaining capacity of these nodes between all flows adjacent to them. Then we "remove" these flows from the network and reduce all node capacities by the bandwidth consumed by the removed flows. In the next iteration, we identify the "next level" bottleneck nodes of the reduced network and repeat the procedure. The process continues until all flows are assigned their fair bandwidth. The details of the algorithm are given in the following pseudocode fragment, which also includes the case of constrained flows.

Procedure *GlobalComputeMMF*

Input: $G(N,L)$, F, $P = \{0 \le \rho_f < \infty : f \in F\}$

Output: maxmin rate vector $R_{mmf} = [r_f^m : f \in F]$ where m is the number of iterations for the algorithm to converge.

Initialization:

i=1, $U_n^0 = 0 \quad \forall n \in N$, $r_f^0 = 0 \quad \forall f \in F$, $F^1 = F$, $N^1 = N$

$$C_n = \begin{cases} 1, & \text{if graph is bipartite} \\ 2/3, & \text{otherwise} \end{cases}, \ \forall n \in N$$

repeat

1. f_n^i = number of flows $f \in F^i$ adjacent to node n

2. $K_1 \quad \min_{n \in N^i} \left(\frac{C_n \quad U_n^{i\ 1}}{f_n^i} \right)$, $K_2 = \min_{f \in F^i} (\rho_f - r_f^{i-1})$

3. $dr^i = \min\{K_1, K_2\}$

4. $B^i = \left\{ m : m = \arg\min_{n \in N^i} \left(\frac{C_n - U_n^{i-1}}{f_n^i} \right) \right\}$

5. $\hat{F}^i = \begin{cases} \left\{ f : f = \arg\min_{j \in F^i} (\rho_j - r_f^{i-1}) \right\}, & K_1 > K_2 \\ \{ f : f \text{ is adjacent to every } n \in B^i \}, & K_1 \le K_2 \end{cases}$

6. $r_f^i = r_f^{i-1} + dr^i, \ \forall f \in F^i$

7. $U_n^i = \sum_{f \text{ adjacent to } n} r_f^i$

8. $N^{i+1} = \{ n : C_n - U_n^i > 0 \}$

9. $F^{i+1} = F^i - \hat{F}^i$

10. *i=i+1*

until (F^i is empty)