

# Providing Quality of Service via Opportunistic Splitting using Stochastic Approximation

Vinay Joseph, Vinod Sharma and Utpal Mukherji  
Department of Electrical Communication Engineering,  
Indian Institute of Science, Bangalore, India.  
Email: {vinay,vinod,utpal}@ece.iisc.ernet.in

**Abstract**—We consider the problem of wireless channel allocation to multiple users. A slot is given to a user with a highest metric (e.g., channel gain) in that slot. The scheduler may not know the channel states of all the users at the beginning of each slot. In this scenario opportunistic splitting is an attractive solution. However this algorithm requires that the metrics of different users form independent, identically distributed (*iid*) sequences with same distribution and that their distribution and number be known to the scheduler. This limits the usefulness of opportunistic splitting. In this paper we develop a parametric version of this algorithm. The optimal parameters of the algorithm are learnt online through a stochastic approximation scheme. Our algorithm does not require the metrics of different users to have the same distribution. The statistics of these metrics and the number of users can be unknown and also vary with time. Each metric sequence can be Markov. We prove the convergence of the algorithm and show its utility by scheduling the channel to maximize its throughput while satisfying some fairness and/or quality of service constraints.

**Keywords:** Quality of Service, Opportunistic Scheduling, Opportunistic Splitting, Stochastic Approximation, Multiple access channel.

## I. INTRODUCTION

In a wireless network often the optimal policy chooses a node with the highest metric. The nodes may be distributed in space and the value of the metric for a node may be available only with the node. For instance, in a wireless Multiple Access Channel (MAC), scheduling a node with the maximum channel gain for transmission maximizes the throughput (provided we perform proper power control) when the nodes have infinite backlogs ([1]). The challenge lies in finding the node quickly using minimum resources.

This problem can be seen as one of decentralized contention resolution (CR) where, in each slot, several nodes want to transmit to a central node and we have to resolve the contention by opportunistically scheduling a node with the maximum value for some metric. A simple scheme is to let all the nodes feed back their metric to the central node following which the central node selects the node with the highest metric. But, when the number of nodes is large, the overhead required for the transmission of the control information from each node can become prohibitively large. An early work that addressed this problem was [2] in which a scheme was proposed which uses the idea of *opportunistic splitting*. The use of splitting for CR is not new ([3]). In [2], opportunistic splitting is used to split the set of contending nodes based on their metrics. We

refer to the splitting algorithm given in [2] for a *homogeneous setting* with *iid* metrics as ALGO-QB. In this setting, the channel gains of the nodes have the same distribution and for each node, the channel gains are *iid* from slot to slot. For this algorithm, it is shown in [2] that, independent of the fading distribution and the number of nodes, the average number of mini-slots required to find the best node is upper bounded by 2.51 which is close to an asymptotic lower bound. In a later work ([4]), the same authors have considered a *heterogeneous setting* where different nodes' channel gains have different distributions although they are still independent from slot to slot. But, they consider only one kind of metric that ensures temporal fairness. In the rest of this paper, the terms homogeneous setting and heterogeneous setting are used with the same meaning as described above.

Some other works on opportunistic CR are discussed next. In [5], a scheme called channel-aware Aloha is studied where users randomly transmit packets with probabilities that increase with their channel gain. Similar opportunistic extensions of ALOHA are also considered in [6] and [7]. In [8], a random access based feedback protocol which uses a fixed number of feedback slots for CR is presented. [9] also considers CR, and improves on and analyzes the policies presented in [2]. Further, [9] extends the algorithms for multiple node selection. However, [8] and [9] restrict themselves to the case in which the channel gains of all nodes are assumed to be *iid*. [10] presents algorithms for reducing feedback required for CR for a heterogeneous (and homogeneous) setting using *static splitting*. All these algorithms require the scheduler to know the distributions of the channel gains and the number of users.

In this paper, we propose decentralized opportunistic splitting algorithms whose parameters are learnt using stochastic approximation algorithms. The algorithms do not need the knowledge of distributions or the number of users, and perform well in heterogeneous settings too. In addition the sequence of metrics can form Markov sequences instead of being *iid*. Using these algorithms we also provide certain Quality of Service (QoS) guarantees. Also, the algorithms can track the parameters if they change slowly with time. The tracking ability makes our algorithms very useful in dynamic systems. We prove the convergence of our stochastic algorithms used in learning the parameters, and illustrate this using simulations.

The paper is organized as follows. Section II explains the

problem scenario. In Section III we present our opportunistic splitting algorithms and Stochastic Approximation (SA) algorithms needed to learn the parameters of the splitting algorithms. Section IV proves the convergence of the SA algorithms. Section V provides a few opportunistic scheduling algorithms which use with the opportunistic splitting algorithms to show their utility and versatility. Section VI provides simulation results to show the efficiency of our algorithms. Section VII concludes the paper.

## II. PROBLEM SETTING

We consider  $N$  nodes each of which has data to transmit to a common receiver  $R$  via a shared wireless channel. The system is slotted with each slot being of length  $T$ . Let  $U_k(i)$  denote the metric associated with node  $i$  in slot  $k$ . We consider the cases when  $\{U_k(i), k \geq 0\}$  form an *iid* sequence and when they are Markov. We assume that each node knows its metric at the beginning of slot  $k$  but not of the other nodes. We also assume that the metrics are non-negative.

At the beginning of each slot  $k$ , a node  $i_k^*$  is to be selected based on a metric for transmission during the slot. An important metric of interest is  $U_k(i) = h_k(i)$  where  $h_k(i)$  is the channel gain of the  $i^{\text{th}}$  node in  $k^{\text{th}}$  slot. For this metric,

$$i_k^* = \arg \max_i h_k(i) \quad (1)$$

maximizes the throughput when the nodes have infinite backlog ([1]).

Since all the  $U_k(i)$  may not be known at a central controller,  $i_k^*$  will be found via a CR algorithm. The initial part of a slot is used for CR. Once a node is selected in the CR phase, the rest of the slot is made available to that node for data transmission. We further divide the CR phase into mini-slots. Only small control packets are transmitted during a mini-slot and hence, length of a mini-slot  $T_{MS}$  can be much less than that of a slot. The number of minislots in a frame is  $N_1 = \lfloor \frac{T}{T_{MS}} \rfloor$ . The receiver  $R$  can decode a control packet only if exactly one node transmits in a mini-slot.

In the next section, we develop a stochastic approximation based algorithm for selecting the node with the highest metric.

## III. CR-ALGORITHM

In this section, we present the algorithms BASIC and MKV. BASIC is used when each metric sequence is *iid* while MKV is for Markov metrics. We first present BASIC.

BASIC consists of two parts: BASIC-IS and BASIC-SA. We use BASIC-IS during the CR phase. The parameters  $\theta_m \triangleq [t_m \ \alpha_m^+ \ \alpha_m^- \ \beta_m^+ \ \beta_m^-]$  of BASIC-IS are learnt using the stochastic approximation algorithm in BASIC-SA.

BASIC-IS operates as follows. At the beginning of each slot, the algorithm sets a threshold  $\tau_1$  for the metric and the nodes with metric greater than this threshold transmit a small control packet (containing only the node id) during the first mini-slot. The BS sends the feedback  $f_b \in \{0, 1, e\}$  where 0,1 and  $e$  indicate that in the last mini-slot there was zero, one and more than one transmission respectively. The event  $f_b = 1$  denotes a success, i.e., the base station has received

only one control packet and it will schedule the node that sent the control packet to send the data packet in that slot. Each node listens to the broadcast from  $R$  containing the feedback and decides on the next step of the algorithm.

In the event of a collision (i.e., feedback =  $e$ ) in the first mini-slot, a new threshold  $\tau_2 > \tau_1$  is evaluated for the second mini-slot. The increase in threshold is decided by the parameter  $\alpha^+$  (see line 7 of BASIC-IS). In the following mini-slot, the nodes with their metric greater than the new threshold transmit the control packet. Similarly, if the first mini-slot is idle (feedback = 0), we use the parameter  $\alpha^-$  to reduce the threshold (see line 15 of BASIC-IS). The algorithm stops in a mini-slot when a node has been identified for transmission or when it reaches the end of the slot.

Note that when there is a collision or an idle mini-slot, we obtain a lower or upper bound respectively on the highest metric among the nodes. In BASIC-IS, the variables  $t_{lower}$  and  $t_{upper}$  denote these bounds. Once these bounds are obtained, we then use the parameters  $\beta^+$  and  $\beta^-$  to update the threshold (see lines 9 and 18 of BASIC-IS). Parameter  $t_{max}$  is chosen large enough so that the metrics are less than  $t_{max}$  with a high probability.

We now present BASIC-IS.

### BASIC-IS:

```

00: Initialize: collisioncheck = 0, idlecheck = 0,  $k = 1$ ,  $\tau_1 = t_m$ ,  $t_{upper} = t_{max}$ ,  $t_{lower} = 0$ 
01:  $f_b = (0, 1, e)$  feedback from last mini-slot;
02:  $k = k + 1$ ;
03: while ( $f_b \neq 1$  OR  $k \leq \lfloor \frac{T}{T_{MS}} \rfloor$ ) {
04:   if( $f_b = e$ ) {
05:     collisioncheck = 1,  $t_{lower} = \tau_{k-1}$ ;
06:     if(idlecheck = 0) {
07:        $\tau_k = \tau_{k-1} + \frac{\alpha_m^+}{c_{scale}} \tau_{k-1}$ ;
08:     } else {
09:        $\tau_k = \tau_{k-1} + \frac{\beta_m^+}{c_{scale}} (t_{upper} - \tau_{k-1})$ ;
10:     }
11:   }
12:   if( $f_b = 0$ ) {
13:     idlecheck = 1,  $t_{upper} = \tau_{k-1}$ ;
14:     if(collisioncheck = 0) {
15:        $\tau_k = \tau_{k-1} - \frac{\alpha_m^-}{c_{scale}} \tau_{k-1}$ ;
16:     } else {
17:        $\tau_k = \tau_{k-1} - \frac{\beta_m^-}{c_{scale}} (\tau_{k-1} - t_{lower})$ ;
18:     }
19:   }
20: }
21:  $f_b = (0, 1, e)$  feedback from last mini-slot;
22:  $k = k + 1$ ;
23: }

```

Our algorithm is inspired by ALGO-QB. But, motivated by QoS applications we want an algorithm that does not require  $N$  and the distribution  $F_U^i$  of metric  $U_k(i)$ ,  $i = 1, \dots, N$ , and can also track the changing dynamics of the system. This could

be done via stochastic approximation if the algorithm has a finite number of (possibly unknown) parameters. Also, in our algorithm, our objective (approximately) is to minimize mean CR time (see BASIC-SA) instead of greedily maximizing the probability of success in the next slot as done in [2].

We execute the following projected stochastic approximation algorithm for learning the parameters of the filter  $\theta_n$  in slot  $n$ .

#### BASIC-SA:

$$\theta_{n+6N_{upd}+1} = P_H \left[ \theta_{n+1} - \epsilon \left( \frac{\mathbf{Y}_n^{\theta_n+\delta} - \mathbf{Y}_n^{\theta_n}}{\delta} \right) \right] \quad (2)$$

and  $\theta_m = \theta_{6MN_{upd}+1}$  for an  $M$  with

$$6MN_{upd} + 1 \leq m \leq 6(M+1)N_{upd}$$

for some  $\epsilon > 0$  and  $\delta > 0$ , where the  $i$ th component of the vectors  $\mathbf{Y}_n^{\theta_n+\delta}$  and  $\mathbf{Y}_n^{\theta_n}$  are given by

$$\mathbf{Y}_n^{\theta_n+\delta}(i) = \sum_{m=6(n-1)N_{upd}+i}^{6(n-1)N_{upd}+(i+1)N_{upd}} N_{CR}(\theta_m + \delta \mathbf{e}_i), \quad (3)$$

and

$$\mathbf{Y}_n^{\theta_n}(i) = \sum_{m=6(n-1)N_{upd}+1}^{6(n-1)N_{upd}+N_{upd}} N_{CR}(\theta_m), \quad (4)$$

respectively, for  $i = 1, 2, 3, 4, 5$ ,  $N_{CR}(\theta)$  is the random variable denoting the number of mini-slots required for CR given that  $\theta$  is the value of the filter in that slot, and  $\mathbf{e}_i$  is the standard unit vector in the  $i$ -th coordinate direction. Also,  $P_H$  is the projection onto the constraint set  $H = \{\theta : \mathbf{a}(i) \leq \theta(i) \leq \mathbf{b}(i)\}$  where  $\theta(i)$  denotes the  $i$ -th element of the vector  $\theta$  and the vectors  $\mathbf{a}$  and  $\mathbf{b}$  are decided by using some rough estimates of statistics of channel gain.

Observe that BASIC-SA is a modified version of the classical Kiefer-Wolfowitz algorithm (see [11]). The filter is updated once every  $(5+1)N_{upd}$  slots which are required for obtaining the components of vectors  $\mathbf{Y}_n^{\theta_n+\delta}$  and  $\mathbf{Y}_n^{\theta_n}$ . By using a small  $\delta > 0$ , we obtain the finite difference approximation of the partial derivatives which is then used in the approximation of the gradient. We use the averaged value of observations over  $N_{upd}$  slots (see (3) and (4)) when approximating the partial derivatives in the gradient with the corresponding finite differences.

If the metrics of the nodes can take specific values with a positive probability (i.e., their distributions have a singular component with respect to Lebesgue measure) then two or more nodes can have the highest metric in a slot with positive probability. Then BASIC algorithm will not be able to provide the node with the highest metric.

However, then we can slightly perturb the distributions to make them absolutely continuous. Alternatively (e.g., when the distributions are discrete) one can uniformly randomize

the metrics between two atoms of its distribution.

As for BASIC, algorithm MKV consists of two parts: MKV-IS and MKV-SA. We use MKV-IS during the CR phase of a slot and the parameters of MKV-IS are learnt using the stochastic approximation algorithms in MKV-SA.

The main difference between BASIC-IS and MKV-IS lies in the initialization of the threshold in the beginning of a slot. In the latter, the threshold for the first mini-slot in a slot is set as the threshold used in the last mini-slot of the previous slot. This improves the performance of the algorithm for the metric under consideration since the metric is correlated in time (at least when it is positively correlated which will often happen in our applications).

#### MKV-IS:

00: Initialize:  $\text{collisioncheck} = 0, \text{idlecheck} = 0,$

$\tau_1 = \tau_{prev}, t_{upper} = t_{max}, t_{lower} = 0, k = 1$

Lines 01-23: Same as in BASIC-IS

24:  $\tau_{prev} = \min(\tau_{k-1}, t_{max})$

Let,  $\theta_n = [ \alpha_n^+ \ \alpha_n^- \ \beta_n^+ \ \beta_n^- ]$ . MKV-SA is similar to BASIC-SA except for the fact that we learn only 4 parameters and hence we update the filter every  $5N_{upd}$  slots.

#### IV. CONVERGENCE OF THE STOCHASTIC APPROXIMATION ALGORITHM

In this section, we provide the conditions for convergence of the stochastic approximation algorithms discussed in the previous section using the Ordinary Differential Equation (ODE) method ([12], [11]). Theorem 1 provides the convergence of BASIC-SA.

*Theorem 1: Assume that the metrics for each node form iid sequences. Suppose that  $F_U^i$  is continuous for each  $i$ . Then, for any nondecreasing sequence of integers  $s_\epsilon$ , for each subsequence of  $\{\theta(\epsilon s_\epsilon + \cdot), \epsilon > 0\}$ , there exists a further subsequence and a process  $\{\theta(\cdot)\}$  such that  $\{\theta(\epsilon s_\epsilon + \cdot)\} \Rightarrow \{\theta(\cdot)\}$  as  $\epsilon \rightarrow 0$  through the convergent subsequence, where*

$$\theta(t) = \theta(0) + \int_0^t \bar{g}(\theta(s)) ds + Z(t), \quad (5)$$

$\Rightarrow$  denotes weak convergence and  $Z(t)$  corresponds to errors obtained when projecting  $\theta_n$  onto the constraint set  $H$  (See [11] for details). The function  $\bar{g} = \left( \frac{E[\mathbf{Y}_n^{\theta_n+\delta}] - E[\mathbf{Y}_n^{\theta_n}]}{\delta} \right)$ .

*Proof:* The proof is provided in [15]. It is omitted here due to lack of space.

The theorem says that, by using our algorithm, the filter  $\theta_n$  tracks the ODE given by (5). Thus, asymptotically  $\theta_n$  will be close to a zero of the gradient of expected number of slots required for CR with a high probability when  $\epsilon$  is small (i.e., will be close to a local minimum of the cost function). We have observed via simulations that the function  $\bar{g}$  has several zeros. Hence, proper initialization of the filter is important to obtain good performance.

Then, as mentioned earlier, we need to obtain a metric sequence  $\{U_k\}$  which has a continuous state space. However

now we use a different transformation (from that for BASIC mentioned above) which ensures that the transition function  $F_{\widehat{U}|\widehat{u}}$  of  $\{\widehat{U}_k\}$  is continuous, uniformly over compact sets of  $\widehat{u}$ . Details are available in [15]

We have also proved the convergence of MKV- SA when  $V_k$  forms a discrete state, aperiodic, ergodic Markov chain.

## V. OPPORTUNISTIC SCHEDULING

In this section, we discuss some opportunistic scheduling schemes for which we can use our CR algorithms. We will evaluate the performance of our algorithms when used along with these schemes in Section VI.

Let  $R_k(i)$  denote the rate at which node  $i$  can transmit in time slot  $k$  if it is scheduled. We consider four opportunistic schedulers in the rest of the paper.

### A. Scheduler 1:

We consider the scheduler in [1] with  $U_k(i) = h_k(i)$ :  
Schedule node

$$i_k^* = \operatorname{argmax}_i (h_k(i))$$

in slot  $k$ .

This scheme maximizes the throughput (provided we do proper power control) when the nodes have infinite backlogs. However it can be unfair to users and may provide large delays.

### B. Scheduler 2:

We consider a scheduling scheme given in [13] where we schedule node

$$i_k^* = \operatorname{argmax}_i (R_k(i) + v^*(i))$$

where  $v^*(i)$  are real parameters such that:

- (a)  $\min_i v^*(i) = 0$ ,
- (b) For all  $i$ ,  $P(i_k^* = i) \geq f_i$ ,
- (c) For all  $i$ , if  $P(i_k^* = i) > f_i$ , then  $v^*(i) = 0$ .

This policy maximizes the average system throughput subject to the fairness constraint that each node  $i$  transmits for a minimum fraction of time  $f_i$  on the channel. The fraction  $f_i$  is ensured by providing an offset  $v^*(i)$  in the rates if needed. We learn the parameters  $v^*(i)$  using the stochastic approximation algorithm

$$v_{k+1}(i)^* = v_k(i)^* + \epsilon_{fair} \left( f_i - I_{\{i_k^*=i\}} \right) I_{\{v_k(i)^* > 0\}}$$

where  $\epsilon_{fair}$  is a small positive constant.

### C. Scheduler 3:

We consider another scheduling scheme given in [13]. Often the different nodes are concerned about the rates they get rather than the fraction of time they transmit. Suppose  $\mathbf{r} = (r_1, r_2, \dots, r_N)$  is feasible, i.e., there exists some scheduling scheme using which each node can be provided a minimum average rate guarantee  $r_i$ . It is shown in [13] that the following

policy maximizes the sum rate subject to guaranteeing these rates to the different nodes. Schedule node

$$i_k^* = \operatorname{argmax}_i (\alpha^*(i) R_k(i))$$

where  $\alpha^*(i)$  are real parameters such that:

- (a)  $\min_i \alpha^*(i) = 1$ ,
- (b) For all  $i$ ,  $E \left[ R_k(i) I_{\{i_k^*=i\}} \right] \geq r_i$ ,
- (c) For all  $i$ , if  $E \left[ R_k(i) I_{\{i_k^*=i\}} \right] > r_i$ , then  $\alpha^*(i) = 1$ .

We learn the parameters  $\alpha^*(i)$  using stochastic approximation algorithm

$$\alpha_{k+1}^*(i) = \alpha_k^*(i) + \epsilon_{rate} (r_i - R_k(i)) \left( 1 - I_{\{\alpha_k^* \leq 1\}} I_{\{r_i \leq R_k(i)\}} \right)$$

where  $\epsilon_{rate}$  is a small positive constant.

### D. Scheduler 4:

Schedule node

$$i_k^* = \operatorname{argmax}_i (q_k(i) R_k(i))$$

where  $q_k(i)$  is the queue length of node  $i$  at the beginning of slot  $k$ . This is a *backpressure algorithm* ([14]) where in every slot, a node with the maximum product of rate and backlog is chosen for transmission. This is a throughput optimal policy, i.e., it stabilizes the queues if it is possible to stabilize them by any policy.

If  $\{h_k(i), k \geq 0\}$  are *iid* then the schedulers 1-3 have *iid* metrics and we use BASIC algorithm for CR. If  $\{h_k(i)\}$  form Markov chains then for schedulers 1-3 we will use MKV. For scheduler 4 in both the cases we use MKV.

## VI. SIMULATIONS

In this section, we evaluate the performance of BASIC and MKV.

First, we study the performance of BASIC used with Scheduler 1 for a homogeneous setting where we compare the performance of BASIC against the algorithm ALGO-QB given in [2]. Note that ALGO-QB can be used only in a homogeneous setting.

For simulating BASIC, we use the following setting:  $T = 100ms$  and  $T_{MS} = 4ms$ ,  $\epsilon = 10^{-6}$ ,  $\delta = 0.01$ ,  $N_{upd} = 10$  slots and  $c_{scale} = 25$ . We start with the filter initialized to [2 2.5 2.5 12.5 12.5]. We use  $a=[0.1 \ 0.25 \ 0.25 \ 0.25 \ 0.25]$  and  $b=[100 \ 50 \ 24.75 \ 50 \ 24.75]$ .

Table I provides the average number of slots required for CR. We consider Rayleigh fading channels with mean channel gain 1.253. Of course BASIC does not use knowledge of the  $F_U^i$  and  $N$ , but learns its parameters. The learning of parameters does take a large number of slots. From the table we see that performance of BASIC is close to ALGO-QB at least for  $N \geq 10$ . We have found this to be true for some other parameter settings we tried.

Table II shows the average number of slots required for CR using BASIC with Scheduler 1 for a heterogeneous setting. The channel gains of the nodes have Rayleigh, Rician and Nakagami fading with different parameters. We see that the

TABLE I  
COMPARISON OF BASIC AND ALGO-QB: HOMOGENEOUS SETTING;  
RAYLEIGH DISTRIBUTED CHANNEL GAINS,

N	5	10	15	20	35	50	100
ALGO-QB	2.30	2.38	2.41	2.45	2.46	2.44	2.45
BASIC	2.51	2.42	2.44	2.46	2.50	2.49	2.52

performance of BASIC is comparable (infact a little better) to that in the homogeneous setting presented above.

TABLE II  
PERFORMANCE OF BASIC: HETEROGENEOUS SETTING

N	5	10	15	20	35	50
BASIC	2.16	2.37	2.41	2.48	2.60	2.46

Now, we study the performance of BASIC when used with Schedulers 2 and 3 under homogeneous and heterogeneous settings. The results show that our algorithms work well even with these schedulers that perform opportunistic scheduling subject to fairness or QoS constraints.

For the simulations, we use the following settings. The channel gains are independent from slot to slot and the rates are given by  $R_k(i) = 1000h_k(i)$ . For the homogeneous setting, channel undergo Rayleigh fading with mean channel gain 1. For the heterogeneous setting we again have Rayleigh fading with mean channel gains uniformly distributed in  $[0.5, 1.5]$ . We choose  $\epsilon_{fair} = 10^{-3}$  and  $\epsilon_{rate} = 10^{-5}$ . We use a run length of 100 million slots. We also scale the metrics used by these schedulers by  $c_{metric} = 1000$  so that the metric lies in the range  $(0, 3)$  with high probability. This helps as we can now use the initialization and constraint set  $H$  used for simulations involving Scheduler 1. The remaining settings are the same as used in Table I.

First, we consider performance with Scheduler 2. The average number of slots required for CR using BASIC for homogeneous and heterogeneous settings are shown in Table III. The fractions of time demanded by the nodes are obtained by scaling randomly obtained numbers lying in  $[0.5/N, 1.5/N]$  so that the sum of the demanded fractions is one. We can see that the performance of BASIC is very similar to that seen for the earlier cases. In [4], an algorithm has been proposed for a heterogeneous setting. But, they restrict themselves to one type of metric which performs scheduling similar to Scheduler 2. Our algorithm provides performance similar to that in [4], even though we do not use the knowledge of the distribution of the channel gains. Further we have seen that all the QoS demands are met with allocated fractions being very close to the demanded fractions. In the heterogeneous setting with  $N = 10$ , an allocated fraction of time is never less than 98% of the corresponding demanded fraction whereas with  $N = 50$ , it is never less than 90% of the corresponding demanded fraction. Less allocation of time is due to the slow convergence of the parameters of Scheduler 2 and still, we see that BASIC works well even here. The convergence is slower for large  $N$ . For homogeneous setting, the differences in allocation are less

than 2% even for  $N = 50$ .

TABLE III  
PERFORMANCE OF BASIC USED WITH SCHEDULER 2

N	5	10	15	20	25	50
Homogeneous Setting	2.34	2.44	2.45	2.48	2.48	2.51
Heterogeneous Setting	2.32	2.42	2.50	2.55	2.66	2.57

Next, we consider Scheduler 3. The average number of slots required for CR using BASIC for homogeneous and heterogeneous settings are shown in Table IV. For the simulations, the rate  $r_i$  demanded by each node  $i$  for homogeneous case, was set as 300, 175, 140, 90, 50 and 25 for the cases  $N = 5, 10, 15, 25, 50$  and 100 respectively. We see that the performance of BASIC is comparable to that seen for the earlier cases. Further, the demands of all the users are met with the allocated rates never going below 98% of the demanded rates. Even this error reduces as we run the code longer.

Now, we provide the simulation results obtained by executing MKV for the following two settings:

- (1) Scheduler 1 is used and the channel gain process  $\{h_k(i); k \geq 1\}$  is a Markov chain.
- (2) Scheduler 4 is used and the channel gains are *iid*.

TABLE IV  
PERFORMANCE OF BASIC USED WITH SCHEDULER 3

N	5	10	15	25	50	100
Homogeneous Setting	2.59	2.44	2.45	2.50	2.53	2.55
Heterogeneous Setting	2.44	2.48	2.54	2.56	2.57	2.61

We first consider the case in which we use Scheduler 1 along with MKV. The channel gain random process is a Markov chain for each node taking values in the set  $\{0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4\}$ . Note that the channel gains take values in a discrete set and hence, we have to use the randomization of metrics discussed in Section III. The initial value  $\theta_0$  for the parameter vector is  $\{0.1, 0.1, 0.5, 0.5\}$ . The probability transition matrix is

$$P = \begin{pmatrix} 0.30 & 0.20 & 0.15 & 0.10 & 0.09 & 0.06 & 0.04 & 0.03 & 0.02 & 0.01 \\ 0.20 & 0.30 & 0.20 & 0.15 & 0.10 & 0.09 & 0.06 & 0.04 & 0.03 & 0.02 \\ 0.15 & 0.20 & 0.30 & 0.20 & 0.15 & 0.10 & 0.09 & 0.06 & 0.04 & 0.03 \\ 0.10 & 0.15 & 0.20 & 0.30 & 0.20 & 0.15 & 0.10 & 0.09 & 0.06 & 0.04 \\ 0.09 & 0.10 & 0.15 & 0.20 & 0.30 & 0.20 & 0.15 & 0.10 & 0.09 & 0.06 \\ 0.06 & 0.09 & 0.10 & 0.15 & 0.20 & 0.30 & 0.20 & 0.15 & 0.10 & 0.09 \\ 0.04 & 0.06 & 0.09 & 0.10 & 0.15 & 0.20 & 0.30 & 0.20 & 0.15 & 0.10 \\ 0.03 & 0.04 & 0.06 & 0.09 & 0.10 & 0.15 & 0.20 & 0.30 & 0.20 & 0.15 \\ 0.02 & 0.03 & 0.04 & 0.06 & 0.09 & 0.10 & 0.15 & 0.20 & 0.30 & 0.20 \\ 0.01 & 0.02 & 0.03 & 0.04 & 0.06 & 0.09 & 0.10 & 0.15 & 0.20 & 0.30 \end{pmatrix}$$

We use a run length of 50 million slots. This is enough to obtain steady state values. The other settings are as for BASIC. The results of the simulations showing the average number of slots required for CR for different values of  $N$  are given in Table V. We can see that the performance of the CR algorithm for this setting is comparable to that of BASIC.

Next, we present the simulation results when we use Scheduler 4 along with MKV. For each node  $i$ , the arrival process  $\{X_k(i), k \geq 0\}$  is *iid* where  $X_k(i)$  is the number of packets arriving to node  $i$  in slot  $k$ . All the nodes use the same transmit power of 0.005 units and the rate at which a node can transmit

TABLE V  
AVERAGE NUMBER OF SLOTS FOR CR USING MKV; METRIC-CHANNEL GAIN; CHANNEL GAIN RANDOM PROCESSES ARE MARKOV

$N$	5	10	15	25	50	100
MKV	2.43	2.50	2.57	2.61	2.65	2.68

is given by  $50000 \log(1 + P_{tx}h)$  units per slot where  $P_{tx}$  is the transmit power and  $h$  is the channel gain for that node. For each node there is Rayleigh fading with the channel gain process *iid* with mean chosen uniformly from the interval  $[0.5, 1.5]$ . In each slot, the number of packets  $X_k(i)$  arriving to the queue of each node is a Poisson random variable with mean  $E[X]$  and each packet is of size 40 bits. The rest of the settings including the initialization are the same as in the above example except for the following modification. Instead of using just the value of the product of queue length and rate as the metric, we add a positive offset of 10000 to the product to obtain the metric that is used in the CR algorithms. Note that even with the new metric, we choose only the node with highest product of queue length and rate. This offset is needed because of the specific nature of the metric involved. When the arrival rates are less, in some slots, the product of queue length and rate can become zero for all the nodes and as a result  $\tau_{prev}$  in MKV can decrease to values close to zero. Then it takes a large number of slots for it to increase to normal values. This will badly affect the performance of the algorithm.

The results of the simulation are shown in Table VI. For each  $N$ , the algorithm was simulated for different arrival rates  $E[X]$ . As we can see, for low arrival rates, the average number of slots for CR is much higher as explained above. At high arrival rates, the performance of the algorithm is comparable for all the values of  $N$ . Thus, the algorithm is efficient when it matters most (under heavy load; for low loads one may just use only the channel gains as the metric).

TABLE VI  
AVERAGE NUMBER OF SLOTS FOR CR USING MKV; METRIC- SUM OF AN OFFSET AND PRODUCT OF QUEUE LENGTH AND RATE

$E[X]$	0.05	0.10	0.25	0.50	1.0	1.5
$N = 5, E[N_{CR}]$	11.15	9.00	4.32	2.90	2.99	2.99
$E[X]$	0.05	0.10	0.20	0.30	0.40	0.80
$N = 10, E[N_{CR}]$	8.71	4.70	2.85	2.88	2.90	2.92
$E[X]$	0.01	0.05	0.10	0.15	0.20	0.40
$N = 25, E[N_{CR}]$	11.02	3.01	2.83	2.85	2.86	2.85
$E[X]$	0.01	0.02	0.05	0.10	0.15	0.20
$N = 50, E[N_{CR}]$	8.45	3.72	2.81	2.82	2.82	2.82

We have also studied the tracking performance of these algorithms when the metric distributions and/or number of users are changing with time, (see [15]).

## VII. CONCLUSIONS AND FUTURE WORK

We consider the problem of wireless channel allocation to a user with the highest metric via opportunistic scheduling.

We have generalized the opportunistic splitting algorithm so that it can be used when the metric statistics and the number of users may not be known to the scheduler and the metric of different users have different statistics. We further allow the sequences of metrics to form Markov sequences. The performance of the algorithm is close to the opportunistic splitting algorithms with *iid* metrics and known statistics. In addition it can track the changing system parameters. We have proved the convergence of the algorithm and shown its utility by developing algorithms which maximize channel throughput while guaranteeing fairness and/or certain quality of service. This algorithm can be also used in other scenarios such as antenna/relay selection in cooperative communication.

## ACKNOWLEDGMENT

This work was partially supported by a research grant from Aerospace and Networking Research consortium.

## REFERENCES

- [1] R. Knopp and P. Humblet, "Information capacity and power control in single-cell multiuser communications," *IEEE International Conference on Communications, ICC, Seattle, 'Gateway to Globalization'*, vol. 1, pp. 331–335 vol.1, Jun 1995.
- [2] X. Qin and R. Berry, "Opportunistic splitting algorithms for wireless networks," in *Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies INFOCOM*, vol. 3, March 2004, pp. 1662–1672 vol.3.
- [3] D. Bertsekas and R. Gallager, *Data Networks*, 2nd ed., Prentice Hall, 2006.
- [4] X. Qin and R. Berry, "Opportunistic splitting algorithms for wireless networks with fairness constraints," in *4th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, April 2006, pp. 1–8.
- [5] —, "Exploiting multiuser diversity for medium access control in wireless networks," in *Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM*, vol. 2, March-April 2003, pp. 1084–1094.
- [6] S. Adireddy and L. Tong, "Exploiting decentralized channel state information for random access," *IEEE Trans. Information Theory*, vol. 51, pp. 537–561, 2002.
- [7] P. Venkitasubramaniam, S. Adireddy, and L. Tong, "Opportunistic aloha and cross layer design for sensor networks," in *IEEE Military Communications Conference MILCOM*, vol. 1, Oct 2003, pp. 705–710.
- [8] T. Tang and R. Heath, "Opportunistic feedback for downlink multiuser diversity," *IEEE Communications Letters*, vol. 9, no. 10, pp. 948–950, Oct. 2005.
- [9] V. Shah, "Distributed cooperative relay selection," *M.E. Thesis, Electrical Communication Engineering Department, Indian Institute of Science, Bangalore*, June 2009.
- [10] S. Patil and G. de Veciana, "Reducing feedback for opportunistic scheduling in wireless systems," *IEEE Transactions on Wireless Communications*, vol. 6, no. 12, pp. 4227–4232, December 2007.
- [11] H. J. Kushner and G. G. Yin, *Stochastic Approximation and Recursive Algorithms and Applications*, 2nd ed., New York: Springer-Verlag, 2003.
- [12] V. S. Borkar, *Stochastic Approximation: A Dynamical Systems Viewpoint*, Cambridge University Press, 2008.
- [13] X. Liu, E. K. P. Chong, and N. B. Shroff, "A framework for opportunistic scheduling in wireless networks," *Comput. Netw.*, vol. 41, no. 4, pp. 451–474, 2003.
- [14] L. Georgiadis, M. Neely, and L. Tassiulas, "Resource allocation and cross-layer control in wireless networks," *Foundations and Trends in Networking*, vol. 1, no. 1, pp. 1–144, 2006.
- [15] V. Joseph, "Algorithms for Efficient Energy Management and Data Transmission in Energy Harvesting Sensor Networks," *M.E. Thesis*, Dept of ECE, Indian Institute of Science, Bangalore, June 2009.