Low-complexity distributed fair scheduling for wireless multi-hop networks

Aleksi Penttinen*, Iordanis Koutsopoulos† and Leandros Tassiulas†
*Networking Laboratory, Helsinki University of Technology, Finland
email: aleksi.penttinen@tkk.fi
†Department of Computer Engineering and Communications, University of Thessaly, Greece
emails: {jordan, leandros}@uth.gr

***Abstract*— Max-min fair bandwidth allocation is a meaningful objective whenever the level of user satisfaction cannot be clearly expressed as a function of the allocated bandwidth. In this work, we address the issue of approximating max-min fairness in a wireless network without the requirement for network-wide node coordination and we present a low-overhead greedy distributed algorithm for reaching this goal. The algorithm is based on distributed computation of a maximum weighted matching based on appropriately defined flow weights and subsequent scheduling of link flows in an effort to provide max-min rates to them. An inherent feature of our approach is its immunity to topology changes as well as to flow traffic variations. Our method is shown to outperform significantly the centralized (yet, conservative) algorithm of max-min fair rate computation in general topologies in terms of total resulting throughput, minimum shares and node resource utilization.**

I. Introduction

The emerging diverse suite of applications and the increasing user demand for obtaining premium service quality when using them in wireless links has led to novel perspectives in user satisfaction. Although the classical concept of quality of service (QoS) provisioning based on explicit statement of QoS requirements by each user is well suited for session-based situations with continuous information flow, it requires connection admission control and resource allocation techniques and it involves additional signaling burden. Thus, it may turn out to be problematic in the presence of mobility and wireless medium dynamics. Alternatively, sessions can specify their satisfaction (utility) as a function of the allocated bandwidth. However, the definition of proper utility functions is not feasible in general. In such cases where users do not specify their resource requirements, an intuitive objective is to split the available resource equally to all sessions. Whenever a user cannot utilize a portion of its allocated resource because of a constraint, this should also be distributed among other sessions. This objective is captured by max-min fairness.

Max-min rate fairness can be provided at the medium access control (MAC) or the network layer. At the MAC layer, fairness properties need to be ensured on a link basis, namely for single-hop flows. At the network layer, fair rates must be provided to end-to-end, multi-hop session flows and this clearly encompasses fairness in single-hop flows. The focus of this work is on fairness at the MAC layer.

Existing work on MAC layer max-min fairness can be classified into two categories. The first one refers to single-channel systems with connection-less multiple access, where fairness properties are sought through a random access scheme. In [1], a framework for implementing fairness by maximizing the sum of user utility functions is proposed, which gives rise to distributed contention resolution methods to achieve the desired rates. Max-min fairness arises as an asymptotic case of a special utility function. Another work along the same lines appeared in [2], where max-min fair rates can be achieved by appropriate flow weights based on adaptation of back-off timer. However, the algorithm requires a priori computation of max-min fair rates in order to find the flow weights. More recently, the work in [3] shows that max-min fair rates can be attained in the context of Aloha with appropriate adjustment of the access probability of nodes in a distributed fashion. Although these distributed access methods require minimal coordination between nodes, they suffer from severe bandwidth loss due to unavoidable collisions. Moreover, fairness is guaranteed only in a probabilistic sense and is meaningful only for large enough time scales.

The second category of studies comprises connection-oriented multiple access methods, where fairness is solicited with conflict-free link scheduling methods. The authors in [4] introduced the concept of max-min fair rate allocation and provide a scheduling policy for achieving max-min fair rate allocation for single-hop session flows and time-slotted systems. Each node assigns service tokens to adjacent links in a round-robin fashion and the weight of a link is the minimum of the stored tokens at the two end nodes. At each slot, the set of links that form the maximum weighted matching of the network graph are scheduled for transmission. This step renders the approach centralized. A distributed slot assignment algorithm that approximates max-min fair bandwidth sharing is presented in [5]. The algorithm is based on local adjustments of link rates by reallocating time slots subject to conflict constraints in an effort to track the corresponding distributed fluid algorithm that provably converges to max-min fair rates. The methods of this class require some amount of node coordination, yet they guarantee collision-free access to resources.

Clearly, it would be desirable to devise a method that com-

This joint research was supported by the EuroNGI Network of Excellence, EC.

bines conflict-free scheduling with minimal node coordination and achieves a good approximation of max-min fair rates. The method should ideally rely on readily available local information and should involve minimal signaling load in the network. This is precisely the subject of this paper. We present a low-complexity, low-overhead distributed algorithm for approximating max-min fair rates in a wireless network of general topology. Our algorithm is based on distributed computation of maximum weighted matching of the network graph with appropriately defined link weights. Apart from its simplicity and its low complexity, the algorithm does not require a time frame (albeit, it needs slot synchronization among nodes) and it can be applicable in the presence of arbitrary topology or channel quality variations and flow traffic demand changes.

The computation of maximum weighted matching also arises as the maximum throughput policy in scheduling in switches with packets queued in the switch input (e.g. [6]), where link weights represent the number of packets waiting to be transmitted. Several variations to the basic approach have also been proposed (see [7],[8] and references therein). Being designed for input-queued switches, these algorithms are suited for bipartite graphs and are not amenable to distributed implementation, since they imply inter-port communication and involve steps that require centralized coordination such as node sorting.

The rest of the paper is organized as follows. Section II includes the model and our assumptions. In section III we present a centralized greedy algorithm that constitutes the basis of our approach and in section IV we describe the proposed algorithm. Section V provides numerical results and section VI concludes our study.

II. System Model

We consider a time-slotted system with control and data time slots, where L control mini-slots precede one data slot. The duration of a mini-slot is much smaller than that of a data slot. A general, non-bipartite network topology graph $G = (V, E)$ is assumed, with vertices representing wireless nodes and links between pairs of nodes showing node connectivity. Let N be the number of nodes in the network. Network-wide slot synchronization is assumed.

We adopt the term "flow class" to distinguish among flows that traverse different links. Two or more flows belong in the same flow class if they run on the same link. There exist J flows in the network. Each flow that traverses a link is represented by a directed edge from the link end-node (the transmitter) to the other node (the receiver). Associated with a network topology graph is the network flow graph $G_f = (V_f, L_f)$ with the same nodes as in the topology graph ($V_f = V$) and edges between each pair of nodes with each edge corresponding to a distinct flow between those nodes. We focus only on single-hop flows in this work. A flow is said to be active in a link if it transmits a packet on that link.

We consider nodes that possess a single transceiver, that is, one hardware unit that can be used to set up a distinct communication link. We assume that there exist only primary scheduling conflicts, so that the same node cannot transmit or receive simultaneously in more than one link. Under this assumption, the set of active flows in the network at a specific time instant must constitute a matching of the network flow graph. In presence of secondary conflicts, i.e. when receiving nodes are interrupted if they hear more than one transmission simultaneously, the problem changes considerably. We do not address the issue in this paper. The work in [9] presents a distributed algorithm for constructing a fixed-length TDMA-based link schedule under a certain fairness metric and secondary conflicts.

A single physical layer transmission rate corresponding to a certain modulation level and/or coding rate is used for every flow, so that time shares are mapped to bandwidth shares. Our model is quite generic as it incorporates the cases of: (i) arbitrary, time-varying packet arrival rates of flows and therefore time-varying bandwidth requirements of flows, and (ii) arbitrary, time-varying topology changes, due to inherent volatility of the wireless networks.

III. A simple centralized greedy algorithm

Let each flow transmit at most one packet in each slot. Associate each flow j, $j = 1, \ldots, J$ with a weight $w_j = C^{n_j}$, where $C > J$ is an arbitrary number and n_j is the number of time slots that have elapsed since flow j transmitted a packet. When a packet of flow j arrives to an empty queue and waits to be transmitted, then $n_j = 1$, while if flow j has no packets to transmit, then $w_j = 0$. Scheduling conflicts reflect interference constraints and determine eligible sets of flows that are allowed to transmit in the same slot. Consider the collection of eligible flow sets in slot t and let $\mathcal{I}(t)$ denote the set of indices with each index i corresponding to one such set I_i. This index set in turn depends on the presence of packets at the transmitter as well as on link availability that is affected by topology variations. Each eligible set of flows I_i is referred to as matching set of flows, since it is a matching of the network flow graph.

The algorithm employed by a centralized greedy scheduler is as follows. In an attempt to approximate max-min fair rates, the algorithm selects the matching set of flows with the maximum total weight for transmission at each time slot, namely it selects the set

$$i^* = \arg \max_{i \in \mathcal{I}(t)} \Big(\sum_{j \in I_i} w_j(t) \Big) \tag{1}$$

We now underline some important properties.

Property 1: A flow j that has not been active in the last k slots has absolute priority in being scheduled in the current slot over all flows which have been active at least once during the last k slots.

Property 2: The greedy scheduling approach guarantees a minimum transmission rate of $1/J$ for each flow.

Property 3: The scheduler converges to a finite sequence of matching sets of flows which forms an edge cover of the network flow graph.

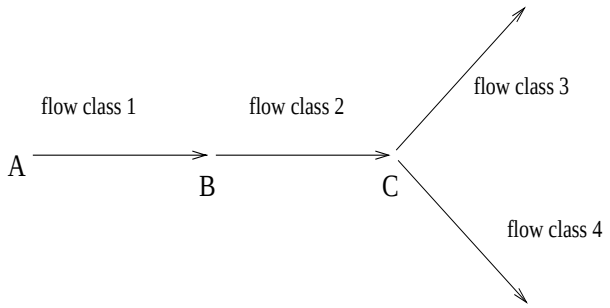

Fig. 1. Illustrative flow graph.

The first property follows from the following line of thoughts. In a certain slot, the minimum weight of a flow that has not received service in the last k slots is C^{k+1} and this occurs in the case that the queue was emptied when it last received service. On the other hand, consider a matching set of flows I that includes only flows that have been served at least once in the last k slots. The weight of I is upper bounded as $\sum_{j\in I} C^{n_j} < JC^k < C^{k+1}$. Thus, the flow that has not received service in the last k slots is granted priority. The second property follows as a consequence of the first one, since in a round of J at most slots, at least one new link will be served. Thus, each every flow will receive service at least every J slots and hence the minimum rate for a flow is $1/J$. The third property follows readily: since $n_j \in \{0,\ldots,J\}$, there exists only a finite number of different weight vectors. At some point after a sequence of slots, the system reaches a vector it has acquired previously and it keeps repeating this schedule (given that possible ties in selecting the matching flow sets are broken in a similar fashion or in a round robin manner). Since all flows are covered, the sequence of matching flow sets forms an edge cover. However, it should be noted that this periodic nature is ensured in the absence of topology or traffic demand variations.

Intuitively, the algorithm serves the flows by attempting to allocate as many links as possible in the same slot, while strict priority is given to flows that are behind others in bandwidth sharing. We illustrate the algorithm in the following topology, borrowed from [4]. There exist four flows and each flow is associated with a different link (flow class). If two links share an end node, the corresponding flows cannot be active in the same slot. The collection of eligible matching sets of flows for scheduling at a certain time instant is $\mathcal{I} = \{\{1\},\{2\},\{3\},\{4\},\{1,3\},\{1,4\}\}$. Assume that all flows have packets to transmit in all time slots. Furthermore, one new packet of each flow arrives at the flow transmitter node at each slot.

Starting with initial values $n_j = 1$ for each flow, the algorithm selects the matching set of flows with the maximal total weight in each time slot. The evolution of the set weights is depicted in the following table, where bold letters denote the matching flow set that is scheduled in each slot. The algorithm results in such a scheduling sequence, that flows receive max-min fair shares.

Flow	Slots						Shares
1	$\mathbf{C}$	$\mathbf{C}$	C	$\mathbf{C^2}$	$\mathbf{C}$	C	$\frac{2}{3}$
2	C	C^2	$\mathbf{C^3}$	C	C^2	$\mathbf{C^3}$	$\frac{1}{3}$
3	$\mathbf{C}$	C	C^2	$\mathbf{C^3}$	C	C^2	$\frac{1}{3}$
4	C	$\mathbf{C^2}$	C	C^2	$\mathbf{C^3}$	C	$\frac{1}{3}$
				repeats			

It should be stressed however that the greedy scheduler does not necessarily attain max-min fair rates. For instance, in the following case with 6 flows and 5 flow matching sets, each corresponding to a row of matrix,

$$\begin{matrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \end{matrix}$$

the greedy approach would select, e.g., rows $1,2,3$ and repeat this pattern, thus yielding rate vector $\frac{1}{3}(2,1,1,1,1,2)$. The max-min fair schedule would use sequence $1,2,3,4,5$ and attain rate vector $\frac{1}{5}(2,2,2,2,2,2)$. Such situations are expected due to the greedy nature of the scheduling rule that does not consider future effects of a decision. In the counterexample above, after using set 3, the algorithm is not aware of the fact that sets 4 and 5 together would yield fairer rates than 1, since the latter appears to be the instantaneously best choice.

IV. Distributed scheduling of single-hop flows

The best known solution to the maximum weighted matching problem in general graphs is of complexity $O(NJ + N^2 \log N)$ [10]. Several approximations have also been developed for the problem that aim either at linear complexity ([11],[12]), or at allowing for distributed implementation [13]. However, these approaches are not amenable to a distributed implementation of scheduling in wireless networks or involve significant burden of control messages. The centralized greedy method in which the edge with the largest weight is sequentially inserted in the matching and all conflicting edges are removed has complexity of $O(J \log N)$ (provided that the edges are sorted a priori). The algorithm results in a matching which has provably a weight of at least $1/2$ times the optimum.

A set of mini-slots preceding each data slot will serve the purpose of control information exchange in our approach. Since control messages themselves are subject to collisions, the control overhead is essentially the number of mini-slots required to exchange coordination information in a distributed *and* conflict-free manner. The proposed algorithm attempts to identify a matching with maximum (or at least, as large as possible) weight in a distributed fashion by using the notion of the greedy scheduler.

The key idea is to give priority to flows (edges) with larger weight. Each node is aware of n_j, the number of timeslots elapsed since last transmission of flow j, for each flow that corresponds to an adjacent edge in the flow graph. This

determines the weight of the edge w_j. Due to the fact that each node locally selects the flow with the largest weight, we can employ as w_j any increasing function of n_j and thus we can assume that $w_j = n_j$ in the sequel. Note however that in the centralized algorithm described in section III, it is crucial for w_j to be an exponential function of n_j, since that ensures the strict global priority for the flows with larger n_j.

The algorithm consists of R iteration rounds. In each round, each node selects the largest-weight incident flow and broadcasts its decision to its neighbors. This procedure takes place for each node in a control mini-slot. The neighbors that receive this information eliminate all other candidate flows destined to or originated from the node that made the decision. If both end-nodes of a flow select the same flow, this flow is added to the matching set of flows. Otherwise, if a node learns that the other end-node of its selected edge has picked another (i.e., heavier) flow, the node becomes idle. The next iteration round is then performed only by the idle nodes. The procedure is referred to as Algorithm 1 and its pseudo-algorithm is as follows.

Algorithm 1 Distributed matching algorithm, input graph $G_f = (V_f, E_f)$ and weights $w_e,\ e \in E_f$

```
M ← ∅ /* matching */
p ← a random permutation of numbers {1, ..., N}.
/* iteration loop */
for i = 1 to R do
    G'_f ← G_f
    for n' = 1 to N do
        n ← p_{n'}
        S_n ← set of edges in G'_f connected to node n.
        if S_n ≠ ∅ then
            e_n ← arg max_{e∈S_n} w_e . Ties are broken randomly.
            w_{e_n} += 0.1
            G'_f ← e_n ∪ (G'_f \ S_n).
            if e_n was already selected by its other end-node then
                M ← M ∪ e_n
                Remove the end nodes of e_n and the attached links from G.
            end if
        end if
    end for
end for
```

The algorithm possesses the following properties:

Property 1: After each iteration round, the resulting assignment is a matching (but not necessarily maximal).

Property 2: At least one link with the maximum weight from the remaining ones is included in the matching after each iteration.

Property 3: The number of iterations $R = N/2$ guarantees a maximal matching.

The first property is obvious from the algorithm. During an iteration round, each node selects at most one of its incident flows, which is then included in set M, provided that the other end-node chooses the same flow. Each node can have at most one attached edge in M and thus M is a matching. In addition, during an iteration loop, a maximum-weight edge will be selected by one of its end-nodes. This increases the weight of the edge and the other end-node will have to select one of its incident maximum-weight edges which have been already selected. Hence, at least one such edge will be included in the matching. This proves property 2 and has direct implications for property 3: since at least one maximum weight edge is included in matching, at least two connected nodes are removed from graph G_f in each iteration loop. After $N/2$ iteration rounds, no connected nodes can remain in G_f.

The first and second property imply that the algorithm can be executed with different values of R to produce feasible matchings. This is a system parameter that captures a trade-off between complexity and efficiency of the matching. The second property is also important, since it provides a bound to the maximum n_j within the greedy scheduler and guarantees a minimum bandwidth, thus enforcing the most important underlying principle of max-min fairness.

A. Implementation issues

Each iteration of algorithm 1 requires that each node broadcast its selection of incident edge to its neighbors so that no collisions occur. A straightforward way of realizing this is via information exchange in a control frame of N mini-slots, in which each node is assigned a unique mini-slot. The order of slots in the frame should preferably be randomized each time, e.g. by using a sequence of pseudo-random numbers that can be stored or computed in each node. For $R = 1$, only one frame of N mini-slots is needed, but if $R > 1$, every other frame will be used by idle nodes to advertise their availability to neighbors. Thus, total control overhead is $(2R-1)N$ mini-slots per data slot. As will become obvious from experimental results, $R = 1$ suffices so that a satisfactory minimum rate is ensured and $R = 2$ is usually enough to achieve maximal matchings and high total rate. It should also be stressed that the amount of overhead is independent of the number of flows.

The algorithm requires that all nodes update the counters n_j for each attached flow. This implies that new arriving flows need to be advertised to the receiver nodes separately. If a flow runs out of packets, this information can simply be piggy-backed onto the last data packet. Furthermore, nodes need to know the length of the control frame in each time slot. This may cause some delay for nodes that appear in the network for the first time, since update in the control frame lenght will be required for nodes entering and nodes leaving the network. The updated length of the control frame must be somehow distributed to the whole network before it can be used.

V. Simulation results

The primary objective of the simulations is to evaluate the performance of the proposed algorithm in terms of different performance metrics and to compare it with some existing approaches. It would also be desirable to quantify the inherent trade-off between control overhead and performance. We

choose to compare our distributed approach (which we refer to as GS for greedy scheduler) to a centralized algorithm to compute max-min fair rates and we refer to this algorithm as MMF. This algorithm provides max-min fair flow rates for non-bipartite graphs under the admittedly conservative constraint that the sum of rates should not exceed $2/3$ for each node [14], [5]. Note that this constraint is a sufficient but not necessary condition to guarantee the feasibility of a rate allocation. MMF does not itself produce a schedule but max-min fair rates that can be achieved by scheduling.

For the simulations we have used random networks with $N = 10$,15,20,25,30 on a unit square with a transmission range of 0.3. For each value of N, results are averaged over 100 network scenarios, in which each node shares two links with its neighbors, one for each direction and each link carries one flow. The average numbers of flows for the different network sizes are $\bar{J} = 18.9$, 46.2, 82.2, 127.0, 185.5, respectively. We run Algorithm 1 in GS for $(R = 1, 2, 3)$ iteration rounds over 1000 time slots.

Figure 2 depicts the total achievable flow rate as a function of number of nodes in the network. This essentially amounts to average number of active links per slot. It can be observed that for $R = 2$, our algorithm outperforms the MMF one by a factor of at least 25% and this performance benefit increases with increasing number of nodes. Similar trends are observed in figure 3, which shows the minimum flow rates in the network. Our approach again outperforms MMF and performs well in terms of providing large enough minimum bandwidth shares to flows, thus enforcing the notion of max-min fairness. Another interesting issue here is that our algorithm exhibits good performance regardless of the value of R, which implies that it can be implemented by using only N mini-slots. Finally, figure 4 shows the average node resource utilization factor. In this case, it can be seen that $R = 2$ is essential in order to guarantee a clear performance benefit over MMF. This is anticipated, since MMF uses the explicit constraint of $2/3$ on node utilization factors for ensuring feasibility of rate allocation. The most important feature of our approach is that it does not restrain resource utilization by posing the aforementioned constraint. Instead, nodes locally control the feasibility of a rate vector by imposing flow preferences. As a result, resources are utilized about 30% more efficiently than in the classical centralized algorithm.

VI. Discussion

We study the problem of approximating max-min fair rates in a wireless network without explicit node coordination and we present a greedy, low-complexity scheduling algorithm that serves this purpose. It was also shown to outperform a centralized (yet, conservative) algorithm of providing max-min fair rates in general topologies. The scheduling discipline is immune to topology or flow traffic variations method and it involves overhead that does not depend on the number of links or flows. When flow traffic demands and wireless link availabilities remain unchanged, the schedule is periodic. However, even in the presence of variations, the scheduled

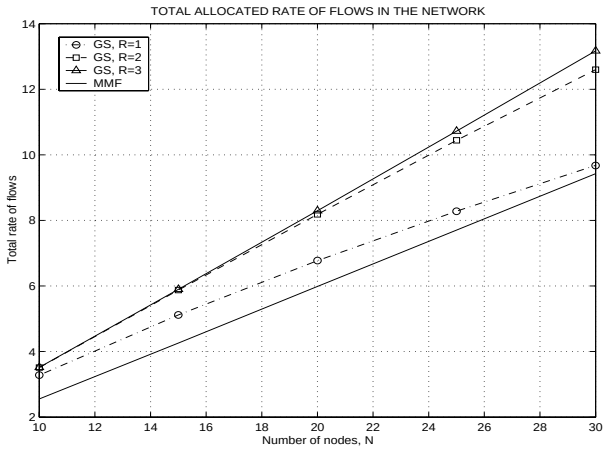

Fig. 2. Total rate of flows versus number of nodes in the network for different scheduling algorithms and values of R.

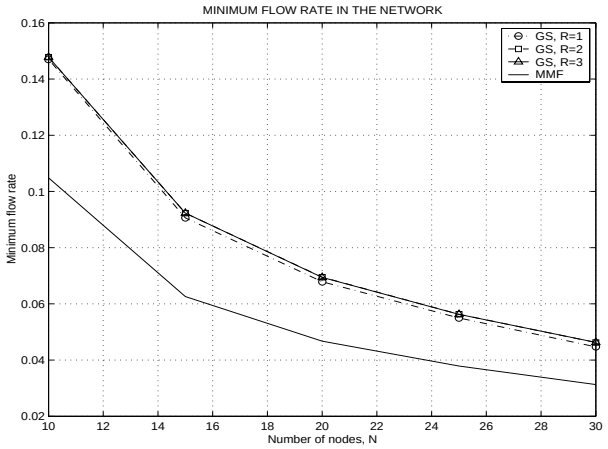

Fig. 3. Minimum flow rate versus number of nodes in the network for different scheduling algorithms and values of R.

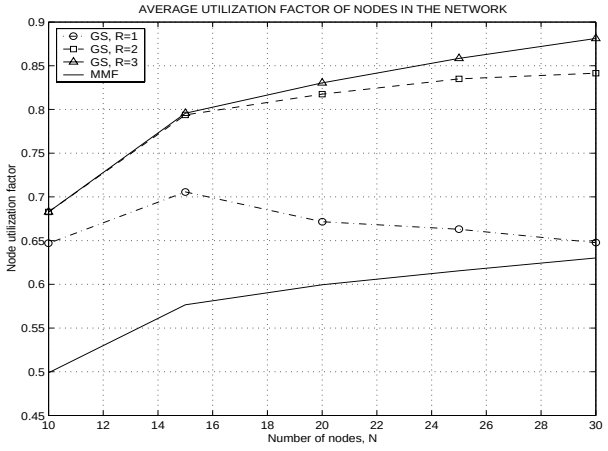

Fig. 4. Average node utilization factor versus number of nodes in the network for different scheduling algorithms and values of R.

flows are continuously adjusted on-the-fly in order to maintain fairness.

There exist several directions for future study. The approach constitutes a first steps towards the goal of fair scheduling in the presence of limited, local knowledge about system status. In that sense, it can be extended to more general resource models, such as that of orthogonal frequency division multiplexing (OFDM) that comprises two-dimensional resource allocation. As another future direction, our approach could also become the initial step for a cross-layer system perspective if combined with distributed algorithms for transmission power adaptation. Then, the impact of fairness provisioning at higher layers on node energy consumption could be assessed in a fully distributed network environment.

REFERENCES

[1] T. Nandagopal, T. Kim,X. Gao and V. Bharghavan, "Achieving MAC layer fairness in wireless packet networks", *Proc. ACM MobiCom,* 2000.

[2] S. Lu, H. Luo and V. Bharghavan, "A new model for packet scheduling in multihop wireless networks", *Proc. ACM MobiCom,,* 2000.

[3] X. Wang and K. Kar, "Distributed algorithms for maxmin fair rate allocation in ALOHA networks", *Proc. Allerton Conference,* 2004.

[4] L. Tassiulas and S. Sarkar, "Maxmin fair scheduling in wireless networks", *Proc. IEEE INFOCOM,* 2002.

[5] T. Salonidis, "Distributed topology organization and transmission scheduling in wireless ad-hoc networks", *Ph.D Thesis,* University of Maryland, College Park, Aug. 2004.

[6] N. McKeown, V. Anantharam and J. Warland, "Achieving 100% Throughput in an input-queued Switch", *Proc. IEEE INFOCOM,* 1996.

[7] N. McKeown, "iSLIP: A scheduling algorithm for Input-Queued Switches", *IEEE/ACM Transactions on Networking,* vol.7, no.2, pp.188-201, April 1999.

[8] V. Tabatabaee and L. Tassiulas, "MNCM a new class of efficient scheduling algorithms for input-buffered switches with no speedup", *Proc. IEEE INFOCOM,* 2003.

[9] I. Chlamtac and A. Lerner, "Fair algorithms for maximal link activation in multi-hop radio networks", *IEEE Transactions on Communications,* vol.35, no.7, pp.739-749, July 1987.

[10] H.N. Gabow, "Data structures for weighted matching and nearest common ancestors with linking", *Proc. 1st Annual ACM-SIAM Symposium on Discrete Algorithms,* pp.434-443, 1990.

[11] D.E. Drake and S. Hougardy, "A simple approximation algorithm for the weighted matching problem", *Inf. Proc. Letters,* vol.85, pp.211213, 2003.

[12] R. Preis, "Linear time $\frac{1}{2}$-approximation algorithm for maximum weighted matching in general graphs", *Proc. STACS,* 1999.

[13] M. Wattenhofer and R. Wattenhofer, "Distributed weighted matching", *Tech. Rep. 420,* Dept. of Computer Science, ETH, 2003.

[14] D. Bertsekas and R. Gallager, *Data Networks,* Prentice Hall, 1992.