On Max-min Fairness and Scheduling in Wireless Ad-Hoc Networks: Analytical Framework and Implementation

Xiao Long Huang
Department of Elec. and Electronic Engineering
Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong, China
eehxl@ee.ust.hk

Brahim Bensaou
Computer Science Department
Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong, China
brahim@cs.ust.hk

ABSTRACT

This paper addresses the problem of fairness in wireless ad-hoc networks. Usually, the problem of fairness in wireless ad-hoc networks is addressed in a classic approach inherited from wired networks. The common assumption is that nodes/flows have pre-assigned fair shares. The task becomes then to modify wired networks' fair queueing algorithms to address wireless networks nature. Wired networks have efficient means of allocating fair shares through admission control and additonally the fair shares remain constant throughout the session duration due to the static nature of the nodes. In ad-hoc networks, it is meaningless to assume statically pre-assigned fair shares, since on one hand, not only the nodes move, but also the routers are mobile, and on the other hand, contention is location dependent, such that in terms of absolute guarantees, fairness would at most mean "avoiding starvation": thus applying rate proportional fair queueing algorithms is beyond the original goal of such algorithm, viz. flow isolation/protection and bandwidth guarantee. In this work we argue in favor of multi-level scheduling for wireless ad-hoc networks with max-min fair allocation of the fair shares at the lower-most layer (MAC layer). This paper, mainly lays down the theoretical framework by which one can calculate the fair shares that would achieve max-min fairness in an ad-hoc network. We design distributed algorithms that allow each node to determine its max-min per-link fair share in a global ad-hoc network without knowledge of the global topology of the network. The results are then used in conjunction with a novel practical scheduling algorithm for IEEE 802.11 to show how fairness is achieved.

Keywords

Ad-hoc networks, Max-min fairness, per-link/per-flow fairness, admission control, scheduling, distributed algorithms

1. INTRODUCTION

In ad-hoc radio networks, due to the limited transmission range of mobile stations, packets ariving from transmitters who may not know of each other may collide at a given receiver, rendering the data unintelligible. This is the so-called "hidden terminal" problem [2], which is known to degrade throughput significantly. Several medium access control protocols have been devised to address this problem (e.g. [1, 2, 3]). Among these, IEEE 802.11 Distributed Foundation Wireless Medium Access Control (DFWMAC) is a proposed standard for wireless ad-hoc and infrastructure LANs. DFWMAC is based on Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) and provides also RTS/CTS access method. The RTS/CTS access method is used to combat the hidden terminal problem by allowing stations to acquire the channel before they transmit the data packets. In other words collisions can occur only during transmission of short control packets (RTS and CTS) rather than during the transmission of potentially very long data packets[1]. Although the RTS/CTS access method can alleviate the effects induced by the presence of hidden terminals, DFWMAC still suffers from a fairness problem that is also induced mainly by the intrinsic multihop nature of ad-hoc networks.

1.1 Fairness in DFWMAC

The fairness problem was first pointed out by Barghavan et al. in [2]. This problem occurs mainly because of hidden terminal problem as well as the backoff scheme used in the DFWMAC protocol. This phenomenon can be simply illustrated by the configuration in Fig. 1.

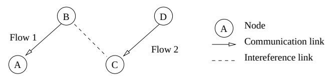


Figure 1: Fairness problem

In this figure, two flows/links[2] compete for the radio resources: flow 1, established between nodes A and B and

[1]As a matter of fact, 802.11 does not solve completely the hidden terminal problem.
[2]In this paper since we are dealing with the link layer, unless otherwise stated, we will use the terms flow and link interchangeably to refer to one-hop communications between two nodes that can hear each other.

flow 2 established between nodes C and D. Due to the nature of the RTS/CTS based protocol, in this configuration, node C overhearing the RTS of node B will not reply to RTSs of node D. Nodes D and B are hidden from each other and thus node B wouldn't overhear node D's RTSs to node C. When the traffic on both flows is heavy, due to the binary exponential backoff used in DFWMAC, node D's contention window would double each time there is collision at C (inferred by D from the absence of CTS from C) until it reaches the maximum value ($CWMax$) specified by the protocol. Meanwhile, node B's window would decrease since B would (eventually) receive ACKs for the data packets sent to node A until it reaches the minimum value ($CWMin$) specified by the protocol. In effect thus node B would have an average contention window size which is much smaller than that of node D and thus would have statistically much higher chance of accessing the channel than node D, which is unfair since the two flows should share the link equally. Simulations have shown that under heavy traffic, the two flows can get throughput in a ratio as high as 3 to 1.

1.2 Recent efforts towards providing fairness

Recently several mechanisms have been proposed to address this problem. Luo et al [4] have proposed a two phases scheduling scheme to achieve fairness in ad-hoc networks. The algorithm constructs a tree comprising all mobile stations, then rearranges the tree so that it becomes conflict-free among the one-hop flows in each level of the tree. During the process of constructing the conflict-free tree, the stations in each level of the tree will propagate the tree knowledge so that the stations can perform a weighted fair queuing (WFQ) scheduling among different levels in the tree. The time required for constructing the tree can however be very long when there are many nodes involved in the network. When mobility is taken into account, say if the root station of the conflict-free tree moves out of its original location, the tree has to be reconstructed to maintain the global fairness among one-hop flows.

Another approach devised by Vaidya and Bahl [5] to address this problem inherits the virtual clock method used in wired networks to provide fair queueing (e.g., [6]). The mobile station broadcasts its virtual clock to its neighboring stations (using piggy backing or in a broadcast channel), and updates its own virtual clock from other stations' broadcasts. Then the mobile station scales its contention window according to the updated virtual clock and its flow's fair share. Simulation results have shown that this approach is good in wireless LAN, in which there are no hidden terminals. However in multi-hop wireless environments, there is a problem among different regions in the network as contention is not homogeneous. Two stations that can interfere with each other while being hidden from each other, may face very different competition (i.e., may hear each a different number of stations). Consequently, the stations will negotiate different virtual clocks from their neighborhood (the one that faces heavier competition will have a slower virtual clock). Thus two stations that can interfere with each other may use different virtual clocks obtained from their different regions, thus, they cannot schedule fairly.

In [7] a measurement-based algorithm is proposed to achieve fairness in ad-hoc networks. The algorithm replaces the binary exponential backoff (BEB) algorithm by another backoff scheme where the contention windows are adjusted according to the stations' fair shares. Each station in the ad-hoc network estimates the amount of traffic it generates against its fair share and the amount of traffic generated by other stations it can overhear against other stations' fair share, and based on this, the station adjusts its contention window size in order to equalize both ratios. This makes the probability of attempting to access the channel proportional to the station's own traffic weight. This algorithm is simple and incurs no additional computational overhead, and the simulation result show how promising it is in ad-hoc environments. However, the algorithm has a major drawback when operating in a dense network where all stations can hear each other's transmissions. That is, the contention window adjustment in [7, 8] replaces the exponential increment of the BEB without paying attention to the traffic and stations density. While the exponential increment mechanism of the BEB of IEEE 802.11 is used to alleviate the frequent collisions when the density of the traffic and stations increases, the stations in the proposed algorithm become very aggressive under the same traffic conditions. Each station overhearing other's successful transmission will reduce it's contention window towards $CWMin$, and on average all contention windows will be much closer to $CWMin$, leading thus to more collisions. While the BEB in the same environment would lead to an average window increasingly closer to $CWMax$ to alleviate the collisions. In short, while the new algorithm addresses very well the fairness problem in ad-hoc networks by overcoming the bad properties of the BEB algorithm, it fails to conserve the good properties of this latter.

In this paper we try to address the problem of fairness more from the point of view of providing a systematic way towards guaranteeing bandwidth and thus some level of quality of service. To understand the fundamental steps of our approach, let us consider the ad-hoc network in Fig. 2 where three end-to-end flows labeled A, B, and C need to be established.

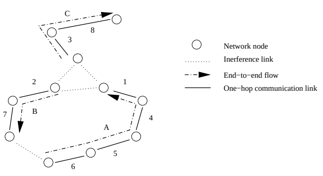


Figure 2: Ad-hoc network configuration

In general, to provide even a semblance of QoS a network needs to be able to decide whether to accept end-to-end flows; it needs to be able to limit new traffic in the network in order not to violate previously admitted flows' QoS, and finally it needs to be able to control and police the flows' generated traffic. In an ad-hoc network based on the minimalist approach of considering only the network layer and not considering any underlying link layer (e.g., [9]), it is simply very tedious if not impossible to achieve these targets. That is because at the network layer,

- nodes have no means to efficiently estimate the bandwidth available to their flows and the flows they route;

- nodes have no means to limit the traffic generated by other nodes in their vicinity;
- a node where an end-to-end flow originates has no means of determining whether it can accept a flow or not, as the bottleneck might be in another node along the route.

Let's assume now that there exists a way to assign fair shares of bandwidth ($\phi_1, ..., \phi_8$) to the single-hop flows (link layer pipelines) numbered 1 to 8 in Fig. 2 and that there exists a method of scheduling that allows us to maintain these fair share of bandwidth at least statistically. Knowing the physical layer capacity, each node would be able to calculate the amount of bandwidth that it has on each outgoing link layer pipeline and thus each node on a given end-to-end routed flow can make the decision whether to accept or reject a flow based only on the flow's bandwidth requirement. Assuming the algorithm used to determine the fair shares is a distributed algorithm that enables collaboration between nodes to maintain the system in a coherent state, nodes in the system can collaborate to prevent, say, a new (non-misbehaving) node from joining, should this otherwise lead to violation of the guarantees already committed.

This paper tries to specify algorithms and mechanisms that would enable nodes with these capabilities. In this paper we focus mainly on designing algorithms to assign fair shares to link layer aggregate flows (pipelines). The criterion we try to achieve is max-min fairness[3]. With this approach, end-to-end flows that compete for the same link level pipe can be scheduled within each pipe according to fair queueing (e.g., deficit round robin [10]). We notably propose a set of novel algorithms to assign max-min fair shares to communication links (one-hop flows) of the global ad-hoc network in a distributed manner, without knowledge of the global topology of the network.

The remainder of this paper is organized as follows. Section 2 discusses and defines max-min fairness in ad-hoc networks. In Section 3 we design a centralized algorithm to calculate max-min fair shares of all the flows of an ad-hoc network from the flow contention graph of that network. Section 4 proposes a new algorithm to assign max-min fair shares in distributed fashion and provides some insight on how this can be implemented in a real network. Section 5 proposes a set of novel protocols which facilitate the implementation of max-min fair scheduling algorithm through a modified backoff scheme for IEEE 802.11 DFWMAC. Section 6 gives some numerical results, and finally, we draw our conclusions and discuss ongoing work in Section 7.

2. MAX-MIN FAIRNESS IN AD-HOC NETWORKS

To minimize collisions, and maximize channel utilization through spacial bandwidth reuse, if two flows are contending flows, they are expected not to be scheduled to transmit simultaneously, otherwise, they should eventually transmit simultaneously in order to maximize network throughput. The contention nature of a global ad-hoc wireless network determines the possible extent of fairness. To ideally achieve max-min fairness in the flow scheduling, we should first be able to allocate fair shares that meet the max-min criterion and then devise a scheduling policy that achieves (if possible) fairness proportionally to these fair shares. To do this, we need to know all contention information. Based on this contention information, node graphs (n-graph) can be converted into flow contention graphs (c-graph). In a flow contention graph, the vertices represent one-hop flows, while the edges represent the existence of contention between the two flows at the ends of the edge. Fig. 3 shows the flow contention graph derived from the network of Fig. 2.

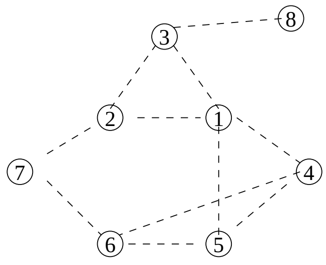


Figure 3: Flow contention graph

Before proceeding any further, we define clearly what this paper is not all about. As mentioned previously, max-min fairness and scheduling are two contradictory objectives, especially in wireless networks. Sometimes, following the max-min fairness definition in [11] we are able to find a set of fair shares that satisfy the conditions in [11] however, when trying to schedule the flows, it is not possible to do so and achieve those fair shares. The most interesting remark is that this is also the case in wired networks. In fact max-min fairness as defined in [11] assumes implicitly that the underlying network is a packet switched network. Feasibility of the rate vector assumes long term average feasibility but not necessarily short term feasibility. In a circuit switched network (either fixed assignment TDMA or FDMA based), it is both necessary and sufficient to have feasibility in short term in order to achieve feasibility in long term. One can construct a counter example where the fair shares vector is feasible according to the algorithm in [11], however, there is no schedule that can achieve th rate vector because of the constraints added by the circuit switched nature of the network. A typical example of this is the case of 5 flows: flow 0, 1, 2, 3, 4, in a 5 routers/link pentagonal subnetwork. Flow i contends with flows $(i-1)$ *mod* 5 on link i and contends with flow $(i+1)$ *mod* 5 on link $i+1$ i.e., 2 contends with 1 and 3, 0 contends with 1 and 4, ..., such that each link is shared by two flows only. Thus the vector of fair shares (1/2, ..., 1/2) is max-min fair according to [11], and there exists indeed a schedule to achieve this should the network be packet switched, however it is easy to see from the flow contention graph that if the network is circuit switched, there is no schedule (partition of resources) that achieves this vector. It is also the case in wireless networks. So the algorithms we propose do not claim to achieve max-min fair scheduling, but rather to achieve max-min fair partitioning of the bandwidth, i.e., the chances to access the channel are allocated

[3]As we'll see later max-min fairness and scheduling can be two contradictory objectives in wireless networks, and some times also in wired networks. In other words, although a set of max-min fair shares exists according to the max-min algorithm, the scheduling according to this set is not necessarily achievable

according to a max-min fair algorithm. If max-min fairness is feasible then it would be achieved. Note also that in our different investigations, the only topology we encountered in which max-min fair partitioning of the bandwidth and max-min fair scheduling are not achieved at the same time is the example above which also can be extended to all polygonal contention graphs with an odd number of vertices. Finally, achieving a set of fair shares that are also schedulable is a combinatorial problem and can be modelled as a graph coloring problem, in which each flow's fair share is proportional to the number of possible colors for the corresponding vertex. In the counter example above, it is clear that 2 colors are not sufficient to color the contention graph thus the fair shares cannot be 0.5.

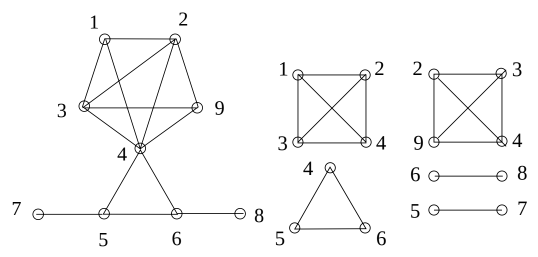
Max-min fairness can be achieved in wired networks among the rates of different sessions sharing a set of links, where a session can span several links and a link can be shared by several sessions. We denote by r_p the rate assigned to session p. The aggregate rate of all sessions on a link a in a network is $F_a = \sum_{p:a\in p} r_p$, i.e. the sum of the rates of all sessions p crossing link a. Let C_a be the capacity of link a, we thus have the following constraints on the vector $r = (r_1, r_2, \dots)$ of allocated rates: $r_p \geq 0, \forall p \in P$ and $F_a \leq C_a, \forall a$. A vector satisfying these constraints is said to be *feasible*. A vector of rates is said to be max-min fair if it is feasible and for each $p \in P$, r_p cannot be increased while maintaining feasibility without decreasing the rate $r_{p'}$ for some session p' for which $r_{p'} \leq r_p$.

Now we want to achieve max-min fair allocation of fair shares in the medium access control in wireless ad-hoc networks. Consider a flow contention graph $G = (N, A)$ where N is the set of flows (vertices) and A is the set of edges of the graph (presence of contention). We define a clique cl [12] as a subset of N such that for all distinct pair $u, v \in cl$, the edge $[u, v] \in A$. If $|cl| = n$ then the clique is said to have degree n.

Fact 1: By construction of the flow contention graph, and the definition of a clique, any two or more flows that belong to the same clique, cannot be scheduled to transmit simultaneously. As a direct consequence of this, we can establish a parallel between wired networks and ad-hoc networks as follows:

- cliques in a flow contention graph are to wireless ad-hoc MAC (one hop flows) what links are to wired networks,
- clique members (vertices) in a flow contention graph are to wireless ad-hoc MAC what sessions are to wired networks.

Since the capacity in a wireless network is function of the space (i.e. can only be characterized by its density over a unit of space), and since the distribution of nodes over space is not known or at least difficult to characterize, it is sufficient to normalize the capacity to 1 for every clique of the contention graph and replace the problem of max-min fair rate allocation by a max-min fair shares assignment. Fig. 4 shows a flow contention graph and the corresponding decomposition into (independent) cliques.



(a) Flow contention graph

(b) Decomposition in cliques

Figure 4: A flow contention graph and its cliques

Max-min fairness in a flow contention graph is thus defined as follows. Given a flow contention graph $G = (V, A)$, denote by $\mathcal{C}$ the corpus of all the cliques of G. Let ϕ_v be the fair share of flow (vertex) $v \in V$. The aggregate fair share allocated over a clique is $\Phi_{cl} = \sum_{v\in cl} \phi_v, \quad cl \in \mathcal{C}$. The vector $\phi = (\phi_1, ...)$ of fair shares that satisfies the constraints, $\phi_v \geq 0 \ \forall v \in V$, $\Phi_{cl} \leq 1, \forall cl \in \mathcal{C}$ is said to be feasible. A feasible vector of shares ϕ is said to max-min fair if for any $v \in V$, increasing ϕ_v cannot be done without decreasing the fair share $\phi_{v'}$ of another flow $v' \in V$ such that $\phi_v \geq \phi_{v'}$.

To allocate the fair shares, it is thus necessary to construct the corpus $\mathcal{C}$ of all the cliques of the contention graph. This problem is well known to be NP-hard [12] and thus working on the global graph of the network would be computationally prohibitive. Our approach is to propose a distributed algorithm run at each node and that works only on a subset of the contention graph which represents only local contentions, requiring thus reasonable computations especially when we know that graph configurations in which the number of independent cliques is very large (e.g., n-partite graphs) are non-natural configurations in the real world of ad-hoc networking. We speculate (not without reason) that in practice, it is most likely for ad-hoc networks based on DFWMAC to have local contention graphs that contain only a few cliques, in which case the problem is solved in a few steps.

3. MAX-MIN FAIR SHARE ASSIGNMENT IN A GLOBAL CONTENTION GRAPH

Let $G = (V, A)$ be the flow contention graph of an ad-hoc network and $\mathcal{C} = \{cl_1, cl_2, ..., cl_n\}$ the corpus of all the cliques of G. Let ϕ_v be the fair share of vertex $v \in V$. Algorithm 1 provides a means to allocate max-min fair shares to all the flows in a contention graph. Note that allocating a fair share does not necessarily mean that there exist a schedule that achieves this set of allocated fair shares.

Fig. 5 shows an example of contention graph, its decomposition in cliques, and the execution of this algorithm. Dashed edges and vertices are those that are removed in step 4 of the algorithm. In the first iteration of the algorithm, the flows of the clique with the smallest ratio c_i/d_i are assigned their

Algorithm 1 Max-min fair share allocation in a global contention graph

```
Init
Let $C = \{c_1, c_2, ..., c_n\}$ be the capacities of the cliques, initialized all to 1;
Let $D = \{d_1, d_2, ..., d_n\}$ be the initial degrees of the cliques;
Define $UCL$ as the set of unfinished cliques, initialized to $\mathcal{C}$;
1: Begin
2: Sort $UCL$ in a non decreasing order of $CD^{-1}$
       $$\frac{c_{i_1}}{d_{i_1}} \leq \frac{c_{i_2}}{d_{i_2}} \leq ... \leq \frac{c_{i_n}}{d_{i_n}}$$
3: Assign a fair share to all vertices in clique $cl_{i_1}$:
       $$\forall v \in cl_{i_1}: \qquad \phi_v = \frac{c_{i_1}}{d_{i_1}}$$
4: Remove all vertices of $cl_{i_1}$ from all other cliques of $UCL$ and update the degrees and capacities of the cliques
       $$\forall cl_k \in UCL: \begin{cases} cl_k \longleftarrow cl_k \setminus cl_{i_1} \cap cl_k \\ d_k \longleftarrow d_k - |cl_{i_1} \cap cl_k| \\ \\ c_k \longleftarrow c_k - \frac{c_{i_1}}{d_{i_1}} \times |cl_{i_1} \cap cl_k| \end{cases}$$
5: if $\max d_i = 1$ then
6:     STOP
7: else
8:     Goto 1
9: end if
10: End
```

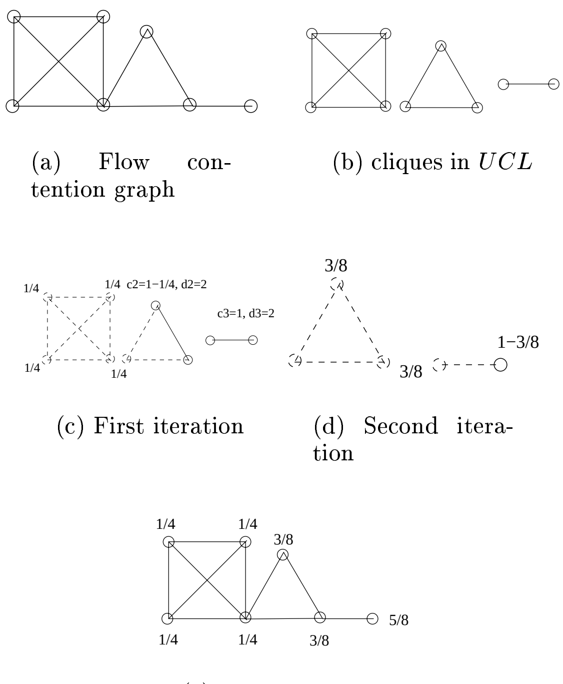
fair shares (in the example, 1/4) and the nodes are removed from all other cliques in UCL (the bottom right node of the square clique is removed from the triangular clique) and the capacity for the altered cliques is adjusted adequately. In the second iteration, the clique with the smallest ratio c_i/d_i (the altered triangular clique, containing now only 2 nodes) is assigned the fair shares and its nodes are removed from the remaining cliques.

Proposition 1. *The fair share allocation obtained when Algorithm 1 finishes verifies the max-min fairness criterion defined earlier.*

To prove this proposition, we introduce another proposition proved in [11]

Proposition 2. *[11] A feasible rate vector r is max-min fair if and only if each session has a bottleneck link with respect to r.*

Given a feasible rate vector r, we say a link a is a bottleneck with respect to r for a session s crossing a, if $F_a = C_a$ and $r_s \geq r_{s'}$ for all sessions s' crossing link a. To prove Proposition 1 it is sufficient to substitute cliques for links, fair share vector for rate vector and flow (vertex) for session



(a) Flow contention graph (b) cliques in UCL

(c) First iteration (d) Second iteration

(e) Final max-min allocation

Figure 5: Max-min fair share allocation example

and prove that the conditions of Proposition 2 hold in this case.

Proof. Due to limited space, and the length of the proof, we give here only a sketch of the proof. Notably, we only establish the premises of a proof by induction. The rest of the proof is left to the reader.

It is easy to see that in the first iteration of the algorithm, the first clique $cl^1_{i_1}$ is a bottleneck for all its vertices, since $\Phi_{cl^1_{i_1}} = \sum_{v \in cl^1_{i_1}} \phi_v = 1$, and, for $v \in cl^1_{i_1}$, $\phi_v \geq \phi_{v'}$, $\forall v' \in cl^1_{i_1}$. In the second iteration, of the algorithm, there exists a clique say cl^1_k which becomes head $cl^2_{i_1}$ of the sorted list of cliques, and $cl^2_{i_1} = cl^1_k \setminus cl^1_k \cap cl^1_{i_1}$. We thus need to show that $cl^2_{i_1}$ is a bottleneck for all of its vertices, i.e., that for all $v \in cl^2_{i_1}$, $\phi_v \geq \phi_{v'}$ for all vertex $v' \in cl^1_k$. We have two sets of such vertices v' those which belong to $cl^2_{i_1}$ and those which belong to cl^1_k but not to $cl^2_{i_1}$. For the first set, using the same argument as previously, we have $\phi_v = \phi_{v'}$. For the second set of vertices v', we have

$$\phi_{v'} = 1/d^1_{i_1},$$

while

$$\phi_v = \frac{1 - n1/d^1_{i_1}}{d^1_k - n}$$

where for simplicity of notations n denotes the number of vertices that belong to both cliques: $n = |cl^1_{i_1} \cap cl^1_k|$. We thus have

$$\phi_v - \phi_{v'} = \frac{1 - n1/d^1_{i_1}}{d^1_k - n} - \frac{1}{d^1_{i_1}}.$$

By virtue of the priority sorting in the first iteration: i.e., $1/d^1_{i_1} \leq ... \leq 1/d^1_k \leq ...$ we have $(d^1_{i_1} - n)/(d^1_k - n) \geq 1$, thus,

$$\begin{aligned} \phi_v - \phi_{v'} &= \frac{1}{d^1_{i_1}} \left(\frac{d^1_{i_1} - n}{d^1_k - n} - 1 \right) \\ &\geq 0, \end{aligned} \qquad (1)$$

Thus in iteration 2 of the algorithm, for all $v \in cl^2_{i_1}$, $\phi_v \geq \phi_{v'}$ for all vertex $v' \in cl^1_k$. And we also have

$$\begin{aligned} \Phi_{cl^1_k} &= d^1_k \phi_v + n\phi_{v'} \\ &= 1. \end{aligned}$$

Which proves that $cl^2_{i_1}$ is bottleneck for all its vertices. Now replacing in the above reasoning iterations 1 and 2 by iterations l and $l+1$ and replacing the capacity and the degrees of the cliques by the adequate values, one can prove more formally by induction the proposition. $\square$

Algorithm 1 of course allows the assignment of max-min fair shares to all flows in the ad-hoc network, however, due to the fact that it is based on the knowledge of the full contention graph of the network, it can only be useful in ad-hoc networks whose MAC protocols use coordinated access such as Blutooth. We can easily imagine each node forwarding to its master its contention information (from what it hears on the network) and the masters of the subnetworks updating their own masters (if any) and so on up to a root of the network which will assign the fair shares and forward them back to the masters who will finally use them to schedule different slave terminals proportionally to their max-min fair shares. In environments that use uncoordinated access such as ad-hoc networks whose MAC protocols use contention (e.g., IEEE 802.11 DFWMAC) or in locally coordinated MAC protocols, this algorithm cannot be used. In the following we provide a distributed algorithm that tries to approximate the behavior of Algorithm 1.

4. DISTRIBUTED MAX-MIN FAIR SHARES ASSIGNMENT

To assign max-min fair shares in multi-hop wireless ad-hoc networks, Algorithm 1 can be executed by each individual node to compute the fair shares of its local flows. We define a node's local flow as a flow that either originates or ends at this node[4]. To ensure the distributed algorithm leads to the same result as the centralized algorithm, two questions arise. First, if the node computes its flows' fair shares based on the local flow contention graph, then what flows should be included in the local flow contention graph? And second, what additional parameters should be exchanged among the nodes in order to give them a compact yet global view of the flow's contention graph? Since we do not give the individual node sufficient topology information, if the node uses Algorithm 1 on its local flow contention graph, the node will often assign non-stable fair share values. So one important parameter for the distributed algorithm is whether an assigned fair share is a stable fair share as given by the centralized algorithm with global topology information or not. From this perspective, Algorithm 2 is derived as follows. For simplicity we use abusively the name normalized capacity of a clique to refer to the ratio c_i/d_i of that clique.

[4]Note that we are addressing the problem at MAC layer, and thus the flows we are manipulating are one hop flows not end-to-end flows.

DEFINITION 1. *Given two cliques cl_i and cl_j, we say that clique cl_j is adjacent to clique cl_i if and only if $cl_i \cap cl_j \neq \emptyset$*

DEFINITION 2. *Let $\mathcal{A}(cl_i)$ be the set of cliques that are adjacent to clique cl_i. We say that a clique cl_i is sub-head clique if and only if, for all clique $cl_j \in \mathcal{A}(cl_i)$, we have $c_i/d_i \leq c_j/d_j$.*

PROPOSITION 3. *The fair share of a vertex that belongs to a sub-head clique cl_i is simply the clique's normalized capacity c_i/d_i*

PROOF. If a clique say cl_k is sub-head we will show that all of its adjacent cliques $(cl_{k_1}, \ldots cl_{k_m})$ are to be assigned their fair shares later than the vertices of cl_k are assigned their final fair share. And thus cl_k can be assigned its fair shares immediately without waiting until it reaches the head of the sorted list. More precisely we will show that anything that is after cl_k in the sorted list does not influence its fair share allocation, and anything that is before does not either because cl_k wouldn't be sub-head otherwise.

Assume there exists a clique cl_i, such that $\mathcal{A}(cl_i) \cap \mathcal{A}(cl_k) \neq \emptyset$, $cl_k \notin \mathcal{A}(cl_i)$ and $c_i/d_i \leq \ldots \leq c_k/d_k \leq c_{k_1}/d_{k_1} \leq \ldots$. When the fair shares of clique cl_i are assigned, all the cliques in $\mathcal{A}(cl_i) \cap \mathcal{A}(cl_k)$ will have their normalized capacity increased in step 4 of Algorithm 1 by virtue of (1) in the proof of Proposition 1. Thus if cl_k is sub-head any allocation of fair shares that occurs changes only the order of the cliques that are adjacent to cl_k with respect to each other but not with respect to cl_k. Assume now that $cl_k \in \mathcal{A}(cl_i)$ in this case cl_k is not sub-head. As a consequence, sub-head cliques are not influenced by other cliques and thus can assign their fair shares as their normalized capacity immediately when they become sub-head. $\square$

As a consequence of Proposition 3, for a node to know whether the fair share allocated to a given vertex v_i is final or not, it needs to know all the cliques that contain v_i as well as all the cliques that have a smaller normalized capacity and that are adjacent to all the cliques that contain v_i. An example of such situation is shown in Fig. 6

Under the commonly made assumption in designing scheduling algorithms for ad-hoc networks, Algorithm 2 should result in the same max-min fair shares allocation to all flows in the network, yet the processing and signalling overhead is much lower than those required by the centralized algorithm (Algorithm 1): in the distributed algorithm messages are exchanged only locally, while in the centralized algorithms

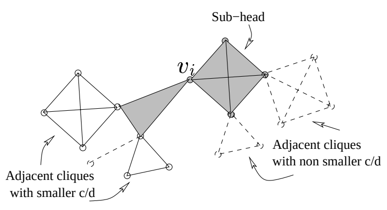
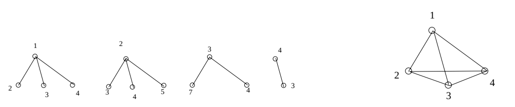

Sub−head
v_i
Adjacent cliques with smaller c/d
Adjacent cliques with non smaller c/d

Figure 6: Example of required partial contention graph

messages are exchanged end-to-end in order to build the global topology; in the distributed algorithm nodes compute only on local much smaller graphs comprising the cliques that contain a flow and the cliques that are adjacent to the cliques that contain the flow. By commonly made assumptions, we mean that the network configuration is made of static or quasi-static nodes, in other words, changes in the topology should occur in a much larger time scale than the time required for the algorithm to converge to a max-min fair allocation. This is not a drawback of our algorithm, its is rather a characteristic of any distributed algorithm that relies on dissemination of information between nodes. We are currently investigating the effects of mobility on this algorithm.

To support Algorithm 2, in computing the fair share of flow v_i, the mobile node should know all the cliques, say CL that contain v_i and for any $v_k \in CL$, $v_k \neq v_i$ the mobile node should know the clique with the smallest normalized capacity that contains v_k.

We define a flow's contention tree as the tree whose root is the flow itself and the leafs are the flows it contends with. To construct the cliques that contain a flow, say v_i, one needs only to know the flow's contention tree and the contention trees of the flows that form the leafs of this tree, as shown in Fig. 7



(a) Flow contention trees: vertex 1 is the local flow

(b) Clique containing flow 1

Figure 7: Local contention graph

PROPOSITION 4. *A node can construct its local cliques if and only if it can obtain the contention tree information from all its neighbors' neighbors.*

PROOF. Let v_i be local flow to a pair of nodes A and B. Let v_m and v_n be local flows to nodes X and Y respectively, and that flows v_m and v_n contend with flow v_i, and v_m contends with v_n. Assuming node X and node Y are not neighbors of node A i.e., A cannot hear any transmission from nodes X and Y. Then necessarily X and Y are neighbors of B and thus A must have obtained the information $(v_i - v_n)$ and $(v_i - v_m)$ from node B. It is easy to see that to complete its cliques, node A needs to get the contention information $(v_m - v_n)$ from either X or Y through B. $\square$

Algorithm 2 Distributed Max-min fair share allocation

```
Init
Foreach node $S$, Let $\mathcal{V}(S)$ be the set of local flows of $S$
$\forall v \in \mathcal{V}(S)$, Let $CL_S^v = \{cl_{S,1}^v, cl_{S,2}^v, ...\}$, such that, $v \in cl_{S,1}^v, v \in cl_{S,2}^v, ...$, and, $\nexists cl_S \in \overline{CL_S^v}$ such that $v \in cl_S$
$\forall v \in \mathcal{V}(S)$, define a message $m_S^v$ as the triplet $\left(cl_S^{v\,*}, \frac{c_{cl_S^{v\,*}}}{d_{cl_S^{v\,*}}}, SHI_{cl_S^{v\,*}}\right)$ where:

  $cl_S^{v\,*}$                              is such that $\frac{c_{cl_S^{v\,*}}}{d_{cl_S^{v\,*}}} = \min_{cl \in CL_S^v} \frac{c_{cl}}{d_{cl}}$;

  $\frac{c_{cl_S^{v\,*}}}{d_{cl_S^{v\,*}}}$  is the normalized (residual) capacity
                                             of clique $cl_S^{v\,*}$; and,

  $SHI_{cl_S^{v\,*}}$                        is a boolean that indicates
                                             whether clique $cl_S^{v\,*}$ is a sub-head clique or not.

Let $\mathcal{M}_S^v = \{m_{S_1}^{u_1}, m_{S_2}^{u_2}, ...\}$ be the set of messages received by node $S$, where $\forall m_{S_i}^{u_i} \in \mathcal{M}_S^v, cl_{S_i}^{u_i\,*} \in \mathcal{A}(CL_S^v)$
 1: Begin  /* Station $S$ executes the following for vertex $v \in \mathcal{V}(S)$ */
 2: for all $m_{S_i}^{u_i} \in \mathcal{M}_S^v$ do
 3:    if $SHI_{cl_{S_i}^{u_i\,*}}$ then
 4:       For all $cl_i^v \in CL_S^v$
              $cl_i^v \longleftarrow cl_i^v \setminus cl_i^v \cap cl_{S_i}^{u_i\,*}$
              $d_{cl_i^v} \longleftarrow d_{cl_i^v} - \left|cl_i^v \cap cl_{S_i}^{u_i\,*}\right|$
              $c_{cl_i^v} \longleftarrow c_{cl_i^v} - \frac{c_{cl_{S_i}^{u_i\,*}}}{d_{cl_{S_i}^{u_i\,*}}} \times \left|cl_i^v \cap cl_{S_i}^{u_i\,*}\right|$
 5:       remove $m_{S_i}^{u_i}$ from $\mathcal{M}_S^v$:    $\mathcal{M}_S^v \longleftarrow \mathcal{M}_S^v \setminus \{m_{S_i}^{u_i}\}$
 6:    end if
 7: end for
 8: $\forall m_{S_i}^{u_i} \in \mathcal{M}_S^v : CL_S^v \longleftarrow CL_S^v \cup cl_{S_i}^{u_i\,*}$
 9: Execute Algorithm 1 on the set $CL_S^v$ to obtain the fair share of $v$
10: construct and broadcast $m_S^v$
11: End
```

Basically, Proposition 4 limits the scope of the local contention graphs that our distributed algorithm will need to work on in order to decide the fair share of a given flow. Roughly speaking, despite the intrinsic NP-hardness of the cliques discovery problem, working on a graph that is driven by contentions only up to a few hops away, is a lot better than working on the graph representing the whole network. This proposition is also the basis for ongoing and future work on proving different properties of the distributed algorithm (such as self-stability, etc.).

5. FAIR MEDIUM ACCESS CONTROL

As an example of application of the algorithms described earlier we choose to design here a fair medium access control protocol based on the standard IEEE 802.11 DFWMAC. The following algorithms as well as the ones designed above would fit in a global architecture for QoS as shown in Fig. 8

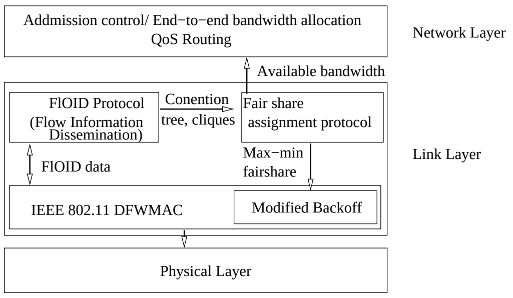


Figure 8: QoS Architecture for ad-hoc networks

In this Figure, a flow information dissemination protocol (FlOID) is in charge of all exchanges of flow contention information between peer nodes up to the scope defined by Proposition 4. Based on the Information collected by FlOID, the fair share assignment protocol computes the max-min fair shares of all flows either originating or ending at the node. Using these max-min fair shares, a novel modified backoff algorithm allows an IEEE 802.11 DFWMAC based access protocol to achieve long term fairness in proportion to the max-min fair shares calculated with Algorithm 1. Due to limited space, we only describe briefly the components involved in this architecture.

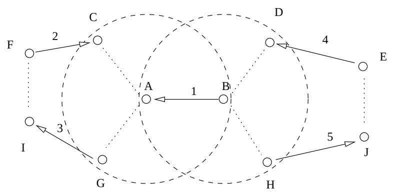
5.1 FlOID

FlOID protocol is in charge of two tasks which lead ultimately to the construction of the local contention graph of the local flows local. In the first step, FlOID builds the local contention tree and then the local cliques to which a given flow belongs. In the second step, FlOID obtains the cliques of all the flows that contend with a given flow.

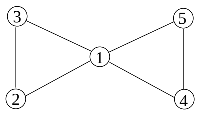
5.1.1 *Construction of local clique information*

In ad-hoc networks, a node can sense partly the flows that contend with its flows either by overhearing the RTS or the CTS. The remaining contending flows can be sensed by the peer node. An example of this is shown in Fig. 9.

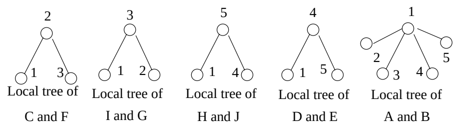
Assume node A in Fig. 9(a) needs to construct the local cliques of flow 1. A can sense flow 2 through the CTSs and ACKs originating from node C to node F, and it can sense flow 3 from the RTSs and data packets originating from node G to node I, so the edges (1-2) and (1-3) are sensed directly by A. Similarly, node B can sense the contentions (1-4) and (1-5). If both nodes at each end of flow 1 exchange their contention information (using either explicit signalling messages or piggy-backing), both A and B would know the local contention tree of flow 1, as shown in Fig. 9(c). On the other hand, nodes C and D know the contention trees of flows 2 and 4 respectively. Thus according to Proposition 4, if node A and node B can receive the contention trees



(a) Ad-hoc networks



(b) Flow 1's conention graph



(c) Local contention trees

Figure 9: Example of contending flows

from node C and node D and their neighbors, node A and node B can construct the local cliques for their local flow 1 as shown in Fig. 9(b).

5.1.2 *Flow information dissemination*

Besides having to construct the local cliques information, each station should obtain all the adjacent cliques of its local clique in order to run Algorithm 1 efficiently. Again, each station can obtain partly the adjacent cliques from its neighbor stations. The remaining part of the adjacent cliques can be obtained from the peer station. For the example in Fig. 9(a), station A can obtain the potentially existing adjacent cliques from its neighbor stations C and G, after stations C and G construct their local cliques for their flow 2 and 3 respectively. Station B can get the potentially existing adjacent cliques from its neighbor stations D and H, after stations D and H construct their local cliques for their flows 4 and 5 respectively. Thus the tasks for disseminating clique information in each station are as follows:

- Send out only the information on the clique that has the smallest normalized capacity among all local cliques, indicating whether it is a subhead upon the current computation or not.
- Forward the available adjacent cliques to the peer node.

5.2 Fair medium access

To schedule one-hop flows in an ad-hoc network in a deterministic pattern, such as in round robin, etc., is difficult because of the hidden terminal problem. One would require to exchange additional information in order to coordinate the nodes. Another possible solution is to adjust the flow's transmission rate according to real time traffic conditions observed by the station. For example if a stations finished workload is T_e and other stations' overheard finished workload is T_o, then the ratio of the two should be maintained proportional the the ratio of their fair shares. This estimation-based dynamic scheduling is very simple to implement statistically for example through backoff schemes.

Based on this concept we propose hereafter two algorithms that attempt at achieving fairness proportionally to the fair shares assigned to the flows. One can achieve per-link fairness by calculating for each link an estimate of its traffic; or per station fairness (as shown in the algorithms below) by achieving the aggregate fair share of all flows originating at a station.

Algorithm 3 Traffic share estimation algorithm

```
Init
Let $\phi_E$ be the aggregate fair share Of flows $V = \{v_1, ..., v_n\}$ originating at this node,
Let $\phi_o$ be the aggregate fair share of flows $V' = \{v'_1, ..., v'_n\}$ sensed by this node $V \cap V' = \emptyset$,
Let $T_e$ be the measured volume of traffic of flows in $V$,
Let $T_o$ be the measured volume of traffic of flows in $V'$,
Let $Last\_Sender$ be a variable used to correlate ACKs and DATA packets for the case where both sender and receiver are in $V'$, this variable is reset periodically if not changed by the algorithm (the period depends on the protocol e.g., in DFWMAC it should be slightly larger than SIFS+propagation delay).
 1: Begin
 2: For each packet $p$
 3: if ($p \rightarrow Destination\_Id == My\_Id$) then
 4:    if ($p \rightarrow Type == ACK$) then
 5:       $T_e := T_e + T_{DATA}$
 6:    else
 7:       if ($p \rightarrow Type == DATA$) then
 8:          $T_o := T_o + T_{DATA}$
 9:       end if
10:    end if
11: else
12:    if ($p \rightarrow Type == ACK$) then
13:       if ($p \rightarrow Destination\_Id \neq Last\_Sender$) then
14:          $T_o := T_o + T_{DATA}$
15:       end if
16:    else
17:       if ($p \rightarrow Type == DATA$) then
18:          $T_o := T_o + T_{DATA}$;
19:          $Last\_Sender := p \rightarrow Sender\_Id$
20:       end if
21:    end if
22: end if
23: End
```

In Algorithm 3, we only estimate successful transmissions or the goodput (i.e., either failed or successful RTSs and CTSs do not count towards the observed share). Since in IEEE 802.11, the possibility of DATA packet colliding is small, we only estimate the volume of traffic according to sensed ACK and DATA packets. Since some flows can be sensed through both DATA and ACK packets, while others may only be sensed through either DATA or ACK packets, a state variable *Last_Sender* is used to correlate Data and ACK packets to avoid duplicate estimation. If the current ACK packet's receiver is the sender of the latest DATA packet sensed, it means both DATA and ACK packet belong to the same transmission. Based on the above estimation result, the station adjusts its contention window to make its flow more or less aggressive by comparing its obtained share to its theoretical share. A fairness reference FR is simply defined as:

$$FR = \frac{T_e/T_o - \phi_e/\phi_o}{\phi_e/\phi_o} \tag{2}$$

Based on the value of FR, the contention window is adjusted according to Algorithm 4, while preserving the BEB. The rational behind preserving the original binary exponential backoff is that, despite the unfairness of the BEB, collisions occur not only because of the hidden terminal problem but also legitimately because of a dramatic increase of traffic density or station density. In this case, the BEB tries to spread the attempts to access the channel over time by increasing the average window size avoiding thus repeated collisions, which would otherwise lead to throughput collapse. In [7], this property is not preserved thus if all stations can hear all other stations in the network, the backoff is too aggressive as the average window size would always be close to $CWMin$. In the present backoff algorithm, a station that succeeds transmission would reset its contention window to a value $CWMin$ that depends on how much share of bandwidth did it obtain.

Algorithm 4 Backoff window adjustment

```
Init
Let $Threshold = 0.05$;
Let $T$ be a given timer period
 1: Begin
 2: Upon each timer period $T$, do,
 3: if ($\phi_o \neq 0 \text{and} \phi_e \neq 0$) then
 4:    if ($T_o == 0 \text{and} T_e \neq 0$) then
 5:       $CWMin := CWMin \times 2$
 6:    else
 7:       if ($Fr \geq Threshold$) then
 8:          $CWMin := CWMin \times 2$
 9:       else
10:          if ($FR \leq -Threshold$) then
11:             $CWMin := CWMin/2$
12:          end if
13:       end if
14:    end if
15: end if
16: End
```

6. NUMERICAL RESULTS

For simplicity, we retained 3 scenarios to show the performance of the fair share based access method under static scenarios. The simulation is done in NS2.1b7. The Mac layer of IEEE 802.11 is modified to include Algorithms 3 and 4. The physical layer emulates the 914MHz Lucent WaveLAN DSSS radio interface with 2Mbps bandwidth, and a propagation range of upto 250m. The MAC layer uses only RTS/CTS access method with CWMin = 31 and CWMax = 1024.

The application generates a stream of packets of 512 bytes each at a constant rate, transported using UDP. The simplest scenario simulated is the two flows scenario shown in Fig. 1, in which the two competing flows should obtain equal share. In Fig. 6, we compare our results to those obtained from the original IEEE 802.11 DFWMAC, where we see the unfairness of this latter.

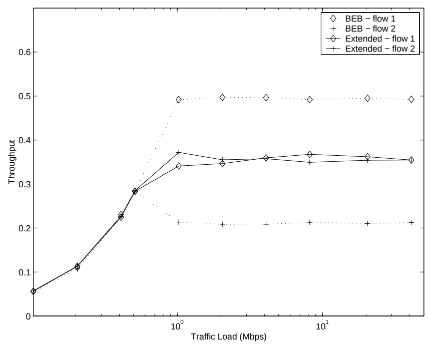
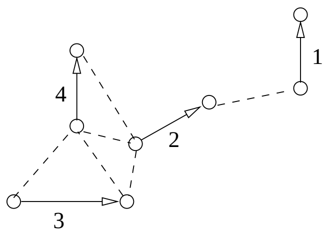
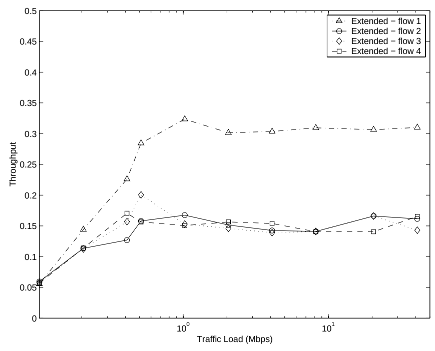


Figure 10: Scenario 1: 4-station scenario

The second scenario we simulated is shown in Fig. 11(a), where stations face unbalanced competition. While flows 3 and 4 compete against 2 flows each, flows 1 and 2 compete against 1 and 3 flows respectively. The throughput achieved by the 4 flows with our algorithm are shown in Fig. 11(b), where as expected, flow 1 obtains twice as much as the other 3 flows.



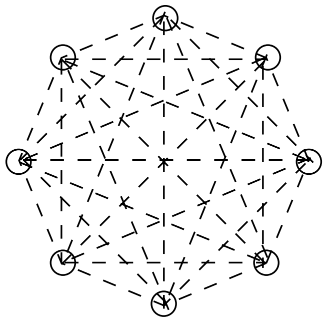
(a) Node graph



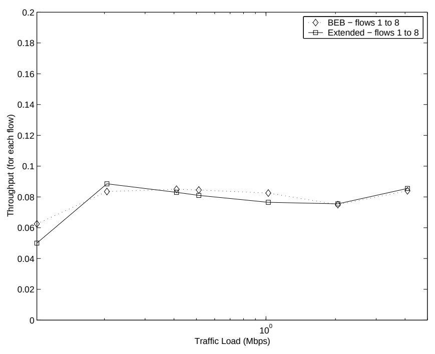
(b) Throughput

Figure 11: Scenario 2: unbalanced contention

Finally to show that unlike [7], our algorithm does not suffer potential throughput collapse when the network is fully connected, the third scenario (Fig. 6) simulates a network of 16 stations, half of which act as senders and the other half as the corresponding receivers. Each station can overhear all remaining 15 stations in the network. Thus we have 8 active flows competing with each other. In this case the flow contention graph is a single clique of degree 8 as shown in Fig. 12(a), and each node is expected to obtain a fair share of 1/8. In this case the original IEEE 802.11 is fair while the backoff algorithm proposed in [7] would achieve fairness but eventually suffer a loss of throughput due to the aggressiveness of the stations in trying to access the channel. Our algorithm as shown in Fig. 12(b) achieves the same throughput per station as does the original DFWMAC with the binary exponential backoff.



(a) Flow contention graph



(b) Throughput

Figure 12: Scenario 3: fully connected symmetric network with balanced contention

7. CONCLUSION AND FUTURE WORK

In this paper we proposed a set of novel algorithms that allow assignment of max-min fair shares to entities competing for shared resources in a distributed manner. Based on these general algorithms, an architecture as well as as a medium access protocol based on IEEE 802.11 DFWMAC layer are proposed to provide max-min fairness in ad-hoc radio networks. The system is shown to support upper layers into providing simply and effectively functions such as admission control (of both flows and terminals) as well as QoS-routing. In the short term, we are focusing on two directions, first implementing the FlOID protocol in NS2, in order to simulate the full architecture in a mobile environment, and second, characterizing theoretically some properties of the distributed algorithm, such as convergence speed etc., in terms of mobility patterns, and other parameters. In a longer term, ongoing work is focused on exploiting topology information gathered in the link layer to implement distributed admission control and QoS-routing algorithms.

8. REFERENCES

[1] IEEE Computer Society LAN MAN Standards Committee, ed., *IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.* IEEE Std 802.11-1997, The Institute of Electrical and Electronics Engineers, New York, 1997.

[2] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, "MACAW: A Media Access Protocol for Wireless LANs," in *Proceedings of ACM SIGCOMM*, 1994.

[3] C. L. Fullmer, *Collision Avoidance Techniques For Packet-Radio Networks.* PhD thesis, University of California at Santa Cruz, June 1998.

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

[5] N. H. Vaidya and P. Bahl, "Fair scheduling in broadcast environments," Tech. Rep. MSR-TR-99-61, Microsoft Research, Dec. 1999.

[6] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single node case," *IEEE/ACM Transactions on Networking*, vol. 1, pp. 344–357, 1993.

[7] B. Bensaou, Y. Wang, and C. Ko, "Fair Media Access in 802.11 based Wireless Ad-Hoc Networks," in *IEEE/ACM MobiHOC*, (Boston, MA.), Aug. 2000.

[8] Y. Wang and B. Bensaou, "Achieving Fairness in IEEE 802.11 DFWMAC with Variable Packet Lengths," in *IEEE Globecom'01 (to appear*, Nov. 2001.

[9] C. Perkins, "On the different mac protocols used with manet routing." Discussions on IETF MANET mailing list, May 2001.

[10] M. Shreedhar and G. Varghese, "Efficient fair queueing using deficit round-robin," *IEEE/ACM Transactions on Networking*, vol. 4, pp. 375–385, June 1996.

[11] D. Bertsekas and R. Gallager, *Data Networks*, ch. 6.5.2, pp. 524–529. Prentice Hall Inc., 1992.

[12] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization, Algorithms and Complexity.* Dover Publications Inc., 1998.