FairMAC: Fair Sharing of Multi-Access Channels in WLAN Hotspots

Prasun Sinha	Yuval Shavitt	Ramachandran Ramjee	Danny Raz	Sneha Kasera
Dept. of CSE	Dept. of EE	Bell Labs	Dept. of CS	Dept. of CS
Ohio State Univ	Tel Aviv Univ, Israel	Lucent Technologies	Technion, Israel	Univ of Utah

***Abstract*— We identify two typical problems in WLAN hotspots that result in unbounded unfairness between upstream and downstream flows. The first unfairness problem arises due to the uniformity of the MAC layer protocol at the access point (AP) and user nodes, that results in equal share to the AP and the user nodes but not to the individual flows. The second unfairness problem arises due to the inability of the physical layer to distinguish frame errors due to hidden terminal based collisions and frame errors due to poor signal strength. We present FairMAC, a deployable solution that addresses these unfairness problems without requiring a change to the 802.11 protocol. Thus our solution is immediately deployable in the millions of currently operational hotspots. We evaluate the performance of our protocol using simulations and a prototype implementation. We show that FairMAC provides fair access to all the flows regardless whether they are originating at the AP or a host.**

I. Introduction

Recent years have witnessed a tremendous growth in number of Wireless LAN (WLAN) hotspots. With increasing number of hotspot users, throughput fairness becomes a critical problem. The problem of fairness in ad-hoc networks [1], [2], [3], [4], [5], [6] and WLANs [7], [8], [9] has been widely studied. However, the problem of unfairness between downstream and upstream flows has received limited attention [7], [8]. Moreover, most of the proposed solutions require changes to the MAC layer making it inapplicable for the millions of currently operational hotspots.

We study two different problems that result in unfairness between upstream and downstream flows. As the 802.11 MAC layer is targeted for per-node fairness, the amount of share of the channel obtained by an AP is independent of the number of users being served by it. As a result, an upstream flow from a user terminal can significantly reduce the downstream flows from the corresponding AP, as illustrated in the following example. Assume that s users are sending data and the AP is transmitting unicast data to r users. Although in this paper, we are assuming that a user is either a sender or a receiver, the generalization where a user can be involved in multiple upstream and multiple downstream flows, is straightforward. We define a flow to be an *ordered pair of nodes sharing the channel*. The fair share, of each flow therefore, is $\frac{1}{s+r}$ of the channel bandwidth. However, in 802.11 the AP competes equally aggressively as the s senders. Thus, the s senders receive $\frac{1}{s+1}$th of the channel bandwidth (assuming $s \geq 1$ and $r \geq 1$, for simplicity of presentation), whereas each of the r receivers receive only $\frac{1}{r(s+1)}$th of the channel bandwidth. The ratio between a sender's share and the fair share is $\frac{s+r}{s+1}$. Therefore, for a constant number of users in a domain, i.e., constant $(s + r)$, the ratio is higher for lower number of senders. A graphical description of the above calculation is presented in Figure 1, where the total number of hosts is 6 and the number of senders is varied from 1 to 5. Observe that if there are a few senders, which is the typical scenario, then the senders can obtain an unfairly large share of the channel. For $s = 1$, we observe that the sender gets $\frac{1}{2}$ of the channel bandwidth, even though its fair share is only $\frac{1}{6}$.

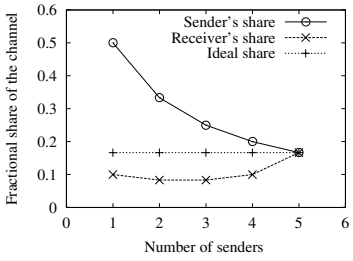

Fig. 1. Channel sharing with 6 hosts, where each node is either a sender or a receiver.

Presence of Hidden terminals and the behavior of IEEE 802.11 MAC protocol can also result in unfairness between downstream and upstream flows. Both collisions due to hidden terminals, and poor transmission channel conditions, show up as bit errors in frames. In 802.11, for both these cases the terminal defers transmissions for a period of time, referred to as EIFS (Extended Inter-Frame Spacing), in order to allow the channel to recover from its error state. We have observed that in typical scenarios hidden terminals may receive control packets at very low power resulting in their less aggressive channel access (due to deferral for EIFS periods) and correspondingly lower channel share.

In this paper we propose a mechanism called FairMAC, to solve the above unfairness problems using a highly deployable approach which does not require any changes to the 802.11 standard. Our approach controls the traffic entering the MAC layer and works above the MAC layer. In particular it can be implemented as a sub-layer directly above the MAC layer. We study the performance of our protocol with simulations and a prototype implementation. FairMAC is also applicable to realistic scenarios with time-varying channel conditions.

The rest of the paper is organized as follows. Section II

0-7803-9428-3/05/$20.00 ©2005 IEEE.

presents two scenarios demonstrating the unfairness problems in WLAN. Section III gives details of the FairMAC algorithm. The performance evaluation using simulations and prototype implementation are described in Sections IV and V respectively. Section VI outlines related work and Section VII concludes the paper with pointers to some open issues.

II. WLAN UNFAIRNESS

Nodes in a WLAN cell may be within or outside the transmission range of each other. For both the scenarios, we show unfairness between upstream and downstream flows.

Scenario I: Mutually in-range nodes: Consider a WLAN hotspot with 6 user nodes, where some users are senders and the rest are receivers. Assume that the flows have backlogged UDP traffic. Figure 2 (based on simulations) shows the total number of packets received for each flow, for varying number of senders. Observe that the graph closely matches the ideal graph shown in Figure 1. We observe that when the number of senders is 0, 5 or 6, all flows get approximately the same share of the channel. For the case with 1 sender and 5 receivers the sender gets half of the bandwidth. Similarly for cases with 2, 3 or 4 senders, we observe that the senders receive higher share of the channel. Observe from Figure 2, that when a sender joins a group of receivers, it takes away half of the bandwidth, and the bandwidth of the receivers, therefore drops drastically. As most of the traffic is typically downstream, such unfairness is common in real WLANs.

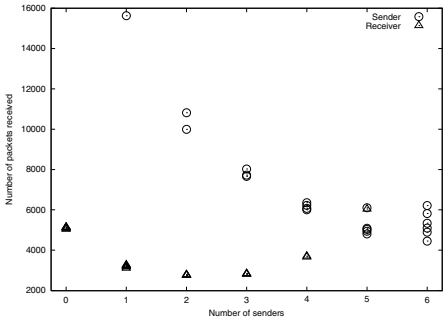

Fig. 2. Scenario I: Unfairness when nodes are mutually in-range.

Scenario II: Mutually out-of-range nodes: Consider a hotspot with two user nodes, that are in transmission range of the AP but out of the transmission range of each other. Suppose node A is sending an upstream traffic (Flow 1) to the AP and node B is receiving a downstream traffic (Flow 2) . Based on simulations, we find that the downstream flow receives approximately 4.3 times (25287 packets as opposed to 5814) as much throughput as the upstream flow. By zooming into a small time interval we can also see the short term unfairness (Figure 3). By analyzing the results, we discover that the observed unfairness is a result of the nodes' inability to distinguish frames in error due to collisions and frames that are received with a very low power. In the scenario described in Figure 3, node A is not in range of node B, and therefore the CTS and ACK from the node B are received at node A with a power level lower than required for a successful reception of a packet. This frame arrives with high bit errors requiring node A to defer transmissions for a period called EIFS (Extended Inter-Frame Spacing). This deferral after a successful transmission of a Flow 2 packet, causes node A to contend less aggressively. However, after a successful transmission of a Flow 1 packet, both node A and the AP will contend equally aggressively. The net result is that Flow 1 gets reduced throughput, and unfair channel access as compared with Flow 2. Although the simulation results are based on typical parameters in *ns-2* such as 250m transmission range and 550m sensing range, the observed unfairness persists over a different region of the WLAN cell if these numbers were to change.

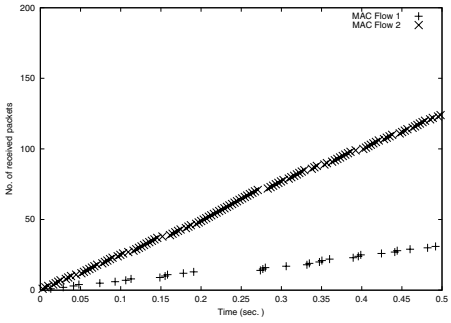

Fig. 3. Scenario II: Unfairness when nodes are mutually out-of-range.

III. OUR APPROACH

We seek to design a solution with the following goals:

- *Per-flow fair:* The solution should provide per-flow fairness, where a flow is defined as a packet stream between a pair of nodes (i.e., AP or user-terminals).
- *Deployable:* It should not require changes to the 802.11 standard to ensure that the solution is highly deployable and immediately applicable to the millions of already operational hotspots.
- *Minimal overhead:* The protocol must minimize the signaling overhead for achieving fairness.

To meet these design goals, our approach FairMAC resides above the MAC layer. It controls the traffic entering the MAC layer by maintaining a rate, to provide per flow fairness. Periodically (every cycle) each node computes the fair share, which is then enforced using a token bucket at the FairMAC layer. A flow which is *satisfied* and whose rate is below the computed fair rate, maintains its previous rate. All other flows use the fair rate. The fair share is therefore, defined as the maximum rate such that the bandwidth is fully utilized, given the constraint that satisfied flows with lower share than the fair share will continue to send at the previous rate. Note that this is same as max-min fair allocation. To provide per flow fairness the AP is required to maintain per flow (by definition of flow, here it means per host in the shared media) queues and maintain separate rates for each queue. Note that the

methodology used in this algorithm and our techniques could be used to achieve other notions of fairness [1], [2] as well.

The pseudo-code for FairMAC is presented in Figure 4. Let n be the number of active (sending/receiving) flows in the network at some given time. The algorithm receives as input the channel capacity, B, the rates of the flows b_i, and whether they are satisfied s_i; it outputs, b_f, the fair rate. This algorithm is executed every *cycle* at each node (hosts and AP). Let $\mathcal{S}$ be the group of satisfied flows and let $\mathcal{U}$ be the group of un-satisfied flows. Then, assuming all the satisfied flows are transmitting below the fair rate, line 4 in Figure 4 gives the fair allocation rate, b_f, since it divides the bandwidth not used by the satisfied stations evenly among the other stations. However, in general, it may be the case that the calculation in line 4 of the pseudo-code will result in some satisfied flows with a rate higher than the fair share. In that case, we iteratively mark the satisfied flows with the highest rate as not satisfied and repeat the calculation (lines 8 - 11 in Figure 4). Though, in theory, this process is linear in the number of flows, in practice it poses no problem since the number of flows is not expected to be large. In cases where this is a concern, one can apply a binary search algorithm for calculating b_f which repeats the calculation only logarithmic (in the number of different flow rates) number of times.

Fair share algorithm

1. Input: $B, \{b_i, s_i\}_{1 \le i \le n}$
2. $\mathcal{S} \leftarrow \{i | s_i = 1\}$
3. $\mathcal{U} \leftarrow \{i | s_i = 0\}$
4. $b_f \leftarrow \frac{B - \sum_{i \in \mathcal{S}} b_i}{|\mathcal{U}|}$
5. if $\forall i \in \mathcal{S} : b_i \le b_f$
6. output b_f
7. else
8. select j s.t. $\forall i \in \mathcal{S} : b_j \ge b_i$
9. $\mathcal{S} \leftarrow \mathcal{S} - \{j\}$
10. $\mathcal{U} \leftarrow \mathcal{U} + \{j\}$
11. goto line 4

Fig. 4. Pseudo-code for fair share computation

In the static case where demands of all the flows are stable, our algorithm computes the fair share b_f using step 4 of the pseudo-code. All flows that require a rate higher than b_f, get the same equal rate of b_f while the rest of the flows get the full amount of their demand. In the dynamic case, where the number of flows (and also their demands) may change, we re-compute the fair share every cycle. Thus, the actual convergence time depends on the choice of length of the cycle. Shorter cycle will speed up the convergence but the computational cost also increases. Thus, we want to keep cycle length in the appropriate range to avoid unnecessary fluctuations and bandwidth under-utilization.

For computing the total bandwidth, several techniques can be used. If some flows are backlogged, then the channel is very likely fully utilized, and the total number of packets sent in the last period is a good approximation of the total channel capacity. The capacity can also be calculated based on the time a packet spends in the MAC layer after passing down from the FairMAC layer. We are currently using the former mechanism and investigating ways to incorporate the latter in order to provide a better and more general mechanism for estimating the channel capacity.

For determining the satisfied or unsatisfied status of a flow, the queue-length can be used as an indicator. If the queue length is beyond a certain threshold, then the node is backlogged and hence should be unsatisfied with its current rate. The satisfied bit can then be sent in the type-of-service field in the IP header. Each node can promiscuously listen to all the traffic (data and MAC control packets) in the WLAN cell, and independently compute the fair share using the estimated bandwidth. However, promiscuous listening is energy inefficient and such independent computations may be inaccurate if packets are not snooped correctly due to channel interference. As all the traffic in the cell goes through the access-point, it can assist in more accurate computation of the fair share. Each AP periodically (every mega-cycle) propagates the fair rate it computes. In a WLAN installation where the served region has higher chances of having hidden terminals, the mega-cycle can not be too long. However if the channel has low interference, the independent computations of the nodes may be accurate and the mega-cycle could be longer. Currently we are using a separate broadcast packet for sending fair share information from the AP every *mega-cycle*. The fair share can be encoded in a few bits to represent discrete bandwidth levels. This information can be piggybacked on the beacons from the AP or other packets transmitted from the AP. Another strategy is to periodically update the state (satisfied bit and rate of every flow) at every node by propagating the information from the AP. The nodes then use this state and then continue updating their local view based on the packets they snoop, till they obtain the next state information (which is more accurate) from the AP. This mechanism allows the nodes to carry on their approximate computation till they receive more accurate information from the AP. However, this mechanism requires the AP to propagate more information than just the fair rate.

Next we describe how to avoid the need for the *satisfied* bit. The key observation is that all flows that are not satisfied, will typically use the same rate, while satisfied flows will have a smaller rate. This is true since all flows run the same algorithm based on the same information from the last iteration. Thus, they all will compute the same fair share. Note that the actual rate of the flows may be smaller than the fair share due to arrival of new flows. However, this will affect all flows, so the fact that all unsatisfied flows will have the same rate still holds. Now, in order to decide which flow belongs to the set S, each station looks at all the b_is and computes the maximum flow b_{max}. All flows within a factor of δ from this value are going to be in U, and all other flows are going to be in S. In other words, we use $S = \{i \mid b_i < b_{max} - \delta\}$, and $U = \{i \mid b_i \ge b_{max} - \delta\}$, in lines 2 and 3 of Figure 4, instead of using the s_i bit. The value of δ should be as small as possible but it should account for variations in throughput due to random channel access. It can also be represented as a fraction of b_{max}.

For each flow, the FairMAC layer needs to maintain a logical packet queue and a token bucket. A host normally has only one flow to the AP, and so it needs to maintain only one packet queue and one token bucket. However, the AP maintains one logical packet queue and one token bucket for each user. To select the particular queue to serve, a round-robin mechanism could be used to check the token buckets of the flows. The rate could be computed in terms of packets/sec or bytes/sec. The token bucket correspondingly has to be in terms of packets or bytes. The token bucket could be implemented as a counter and the value of the counter denotes either the number of packets or number of bytes that may now be sent to the MAC layer.

FairMAC is highly deployable as it does not require changes to the MAC layer and resides as a sub-layer above the MAC layer. It provides short-term as well as long-term fairness, as packets entering the MAC layer are rate controlled at every node sharing the channel. FairMAC does not require changing any packet formats. The optional information that needs to be propagated if the AP chooses to send periodic (every mega-cycle) updates to all the other users, only requires a few bits. Thus, FairMAC meets all the design goals listed earlier.

IV. Performance Evaluation: Simulations

We present results from simulations in *ns2* for a single WLAN cell to establish the correctness of FairMAC. The raw channel capacity of 2 Mbps and transmission range of 250 m is used. The height of the token bucket was set to 2. The queue threshold to determine if a node is satisfied with its flow or not, was set to 3 packets. Our experiments were based on UDP flows with a packet size of 512 bytes. The rate compute cycle and the rate broadcast mega-cycle were set to 0.1s and 0.5s respectively.

Scenario I: Mutually in-range nodes: Figure 5 shows that by

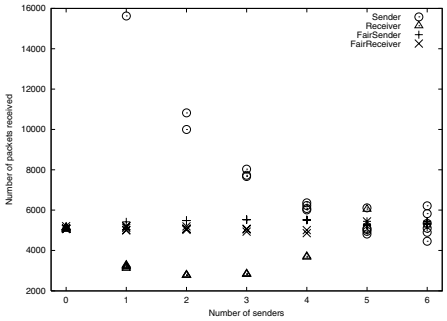

Fig. 5. 6-node scenario: nodes are in range of each other

using FairMAC, all the senders and receivers receive almost the same share of the channel across all cases, i.e., for varying number of senders. Based on promiscuous listening all the nodes learn of the 6 flows trying to access the channel. Nodes also learn that all the flows are *unsatisfied*, as they are backlogged. Therefore the channel capacity computed is divided equally (see algorithm description in Section III) and our FairMAC layer constrains the packet flow to the fair share. In a typical scenario, we would expect most of the users to be downloading data and only a few sending data. This would correspond to the few senders and more receivers scenario in our experiment which leads to heavy unfairness in current systems using IEEE 802.11 (see unfairness for number of senders = 1 or 2 in Figure 5).

Observe that the sender(s) usually get slightly higher share of the channel. This effect is because of our approach of achieving fairness by mechanisms above the MAC layer, without modifying the MAC protocol. Once the MAC layer receives a packet from our FairMAC layer, the FairMAC layer loses control of the packet, and the packet is (re)transmitted according to the MAC protocol. Therefore, forcing the MAC layer to support a high rate is comparatively difficult than forcing a lower rate. Thus, for multiple receivers, the AP has to send data at a much higher rate than the senders and this leads to the AP not being able to exactly acquire its fair share.

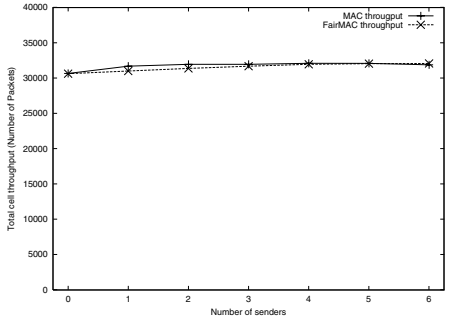

Fig. 6. 6-node scenario: total throughput

Adding a pacing layer above the MAC layer might cause the network to be under-utilized as packets may be blocked at the FairMAC controller. It turns out that this does not happen in FairMAC. Figure 6 depicts the total throughput of the wireless LAN with and without FairMAC for the above scenario. As can be observed, adding FairMAC does not change the total throughput, regardless of the number of senders.

Scenario II: Mutually out-of-range nodes: Based on a 100

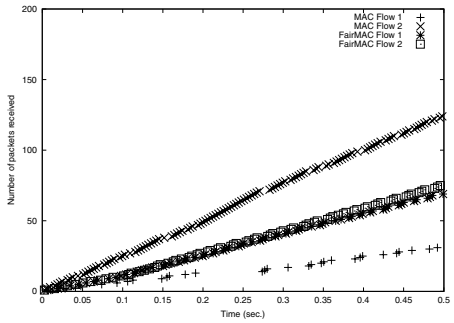

Fig. 7. 2-node Scenario: nodes are out of range of each other

sec of simulation we observe that the uplink flow (Flow 1, A→AP) receives 5814 packets whereas the downlink flow (Flow 2, AP→B) receives 25287 packets which is higher by

a factor of 4.3. In case of FairMAC, the two senders, namely the AP and the node A learn from promiscuous listening about each other, compute the available bandwidth, and share the bandwidth among them as both the flows are *unsatisfied*, i.e., both the flows are backlogged (see Section III). Thus, the flow of packets entering the MAC layers of the AP and node A are constrained to occupy only the fair share of the channel. In case of FairMAC, the A→AP flow receives 14534 packets and the AP→B flow receives 15999 packets. In case of FairMAC, the share of the AP→B flow is always slightly higher than A→AP flow, which is a result of the MAC behavior presented in Section II, but its effect is highly reduced by FairMAC due to controlled rate of packets arriving at the MAC layer. By zooming into 0.5 sec of simulations (Figure 7), we observe that the unfairness is not only long term but also observed over very short term. The figure also shows that the flows in FairMAC are well interleaved which leads to short term fair channel access which is missing in case of the regular MAC.

V. Performance Evaluation: Prototype Implementation

This section presents the results from experiments on our test-bed for understanding implementational issues, and demonstrating the benefits of FairMAC. The implementation requires two main components: a policer that sits in the IP layer and limits the throughput of outgoing packets and a snoop agent that monitors and sets the fair share bandwidth value.

We used the Dummynet system (www.dummynet.com) for implementing the token bucket. Dummynet is a system utility that permits the control of traffic going through various network interfaces, by applying bandwidth and queue size limitations. This is implemented using a mechanism called a "pipe". A Dummynet pipe is characterized by bandwidth, delay, queue size, and loss rate, which can be configured with the *ipfw* program. For our snoop agent, we modified Tcpdump by adding a new FairMAC algorithm option. Tcpdump is a program that allows us to snoop packets from the network interface card efficiently using the Berkeley Packet Filter. Using the new option, Tcpdump calculates the host's fair share bandwidth by monitoring traffic promiscuously on a specified interface. At the end of each cycle, if the algorithm determines that the host's monitored bandwidth and the host's calculated fair share bandwidth differs, then Tcpdump invokes the *ipfw* program and sets the calculated fair share value as the new bandwidth for the outgoing pipe.

In our test-bed one PC serves as a base station, three other PCs serve as mobile hosts, and another PC serves as a fixed host. The mobiles connect to the base station through a 2Mbps wireless LAN while the fixed host is connected to the AP through a 10Mbps Ethernet. In our experiments, the mobile hosts and AP were in range of each other and thus did not experience the hidden terminal problem. Also, all the measurements were obtained on 333MHz PCs running the FreeBSD 3.1 operating system. Figure 8(a) depicts the instantaneous throughput of the mobile senders and receivers

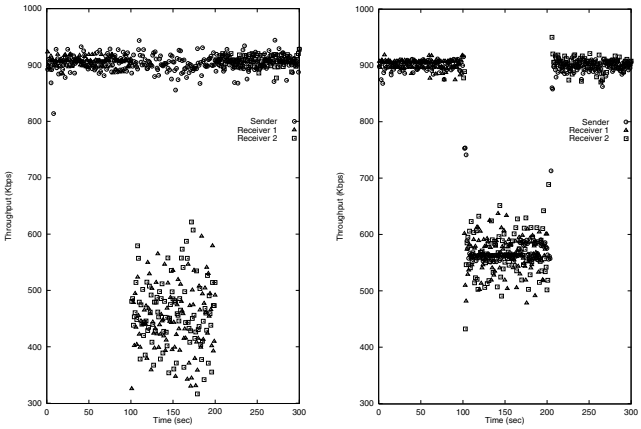

Fig. 8. Throughput vs time (a) without FairMAC; (b) with FairMAC

Mobile host	Without FairMAC			With FairMAC		
	1-100s	101-200s	201-300s	1-100s	101-200s	201-300s
Sender	903	902	906	900	575	897
Rcvr 1	906	452	0	903	562	0
Rcvr 2	0	461	907	0	572	903

TABLE I
Avg. Throughput (in Kbps)

versus time when our FairMAC algorithm is not running. At time 0, we start with one sender and one receiver; we see that the sender and receiver achieve similar throughput of around 900Kbps (see Table I). At time 100 seconds, we add another mobile receiver into the system for a total of two receivers and one sender. It is clear that the two receivers get a much smaller share of the bandwidth of around 450Kbps while the sender continues to achieve higher throughput of around 900Kbps. At time 200 seconds, one of the receiver departs and we find that the system becomes fair again with the sender and receiver sharing the same throughput.

Figure 8(b) shows that the throughput for FairMAC. Between 0 to 100s and 200 to 300s, the throughput is nearly 900 Kbps for all the flows (also see Table I). At time 100 seconds, when a mobile receiver is added, all the flows get similar bandwidth of around 570Kbps. Also, note that the FairMAC algorithm is able to ratchet the sender's rate down to the fair value very fast (a couple of seconds) and maintains that value in a stable manner. Another side benefit to achieving a fair share of the multiple-access channel is reduced variability. Note that the variability of the receivers is much less in Figure 8(b) as compared to receivers in Figure 8(a) between time 100 and time 200 seconds. At time 200 seconds, when one of the receiver departs we find that FairMAC again reverts the senders back quickly to the higher fair share value of around 900Kbps. Thus, the FairMAC algorithm is able to dynamically maintain a stable and fair share access to the wireless multiple-access channel.

VI. Related Work

Most solutions for solving unfairness issues in wireless networks require changes to the MAC layer. Deng and Chang [10] suggested to change the back-off period according to the station's priority. The lower the priority the higher is the

maximum back-off period. Barry *et al.* [11] followed this line and suggested to use two distinct back-off periods for two priority classes. Vaidya *et al.* [1] suggested a distributed algorithm that calculates the back-off period for the stations such that the resulting access to the channel will closely match the Self-Clocked Fair Queuing (SCFQ [12]) scheduling. Aad and Castelluccia [13] suggested three differentiation mechanisms based on scaling of the congestion window, modifying the IFSs, and changing the maximum frame length.

Lu *et al.* [8] were the first to identify the problem of fairness among users in a wireless LAN due to the different role of the AP. However, their solution, to the problem was a centralized scheduling algorithm to be performed at the AP. In addition, their solution required a special MAC algorithm where slots for transmission are specifically allocated to the other stations based on scheduling algorithm. Nandagopal *et al.* [2] suggest a fairness model that also identifies the difference between node fairness and flow fairness. The model is used to compare the fairness achieved by different back-off mechanisms. Pilosof et. al. [7] study unfairness among TCP flows in wireless LANs using real experiments, simulations and analysis. TCP's closed loop control and the IEEE 802.11 MAC together are found to cause high unfairness between upstream and downstream flows, especially when the buffer at the base-station is small.

Another line of research [14] suggests to employ bandwidth reservation in order to support quality of service (QoS). This approach is suited for flows that have specific QoS requirements, but cannot be used for fair sharing of flexible flows, such as IP best effort traffic and ATM ABR traffic. In addition, the method fails to integrate best effort and QoS traffic in the same MA channel. Using our proposal for the best effort traffic can, thus, augment this standardization effort [14].

Koksal *et al.* [3] noticed problems with short term fairness in CSMA/CA systems. We show here that long term problems exist as well, and our algorithm is shown to preserve fairness in both short and long term. The unfairness resulting from the hidden terminal in multi-hop wireless networks is also addressed by solutions for max-min fairness using MAC layer modifications by several papers [4], [5], [6], [9]. To the best of our knowledge the specific problem of WLAN unfairness in upstream and downstream UDP flows, and solutions that do not require changes to the MAC layer were not studied before. In [15], the authors report unfairness problems in 802.11 based networks that occur due to differences in signal strength. Fair scheduling algorithms that consider the effects of channel errors [16], [17] require changes to the MAC layer.

VII. Summary and Future Work

We presented two typical scenarios that can result in unbounded throughput unfairness between upstream and downstream flows. We proposed a deployable solution which can be implemented as a sub-layer directly above the MAC layer for providing fair channel access. We demonstrated the effectiveness of our protocol with simulations using *ns2* and experimentation in a WLAN test-bed with a prototype implementation, and observed that it provides fair access to the channel.

As part of future work, we plan to study the effect of interference from multiple adjacent cells and external noise sources. Noise introduced from neighboring cells or other sources results in varying available bandwidth at different locations in the cell. For robust bandwidth estimation, the base-station can learn the channel bandwidth estimations from each terminal and use such estimates along with its own estimation, to compute accurate fair shares. As pointed out in Section III, more general techniques are needed for monitoring the available channel bandwidth in real time.

References

[1] N. H. Vaidya, P. Bahl, and S. Gupta, "Distributed Fair Scheduling in a Wireless LAN," in *MobiCom 2000*, Boston, MA, USA, Aug. 2000.

[2] T. Nandagopal, T. Kim, X. Gao, and V. Bharghavan, "Achieving MAC Layer Fairness in Wireless Packet Networks," in *ACM Mobicom 2000*, Boston, MA, USA, Aug. 2000.

[3] C. E. Koksal, H. Kassab, and H. Balakrishnan, "An Analysis of Short-Term Fairness in Wireless Media Access Protocols," MIT, LCS, Tech. Rep. MIT-LCS-TR-807, May 2000, also a short presentation at ACM SIGMETRICS 2000.

[4] L. Tassiulas and S. Sarkar, "Maxmin Fair Scheduling in Wireless Networks," in *Proc. IEEE INFOCOM*, New York, NY, USA, June 2002.

[5] D. Julian, M. Chiang, D. O'Neill, and S. Boyd, "QoS and Fairness Constrained Convex Optimization of Resource Allocation for Cellular and Ad Hoc Networks," in *Proc. IEEE INFOCOM*, New York, NY, USA, June 2002.

[6] Z. Fang, B. Bensaou, and Y. Wang, "Performance evaluation of a fair backoff algorithm for IEEE 802.11 DFWMAC," in *Proceedings of the Third ACM International Symposium on Mobile Ad hoc Networking & Computing*. ACM Press, 2002, pp. 48–57.

[7] S. Pilosof, R. Ramjee, D. Raz, Y. Shavitt, and P. Sinha, "Understanding TCP Fairness over Wireless LAN," in *Proc. IEEE INFOCOM*, San Francisco, CA, USA, Mar. 2003.

[8] S. Lu, V. Bharghavan, and R. Srikant, "Fair Scheduling in Wireless Packet Networks," in *ACM SIGCOMM'97*, Cannes, France, Sept. 1997.

[9] T. Ozugur, M. Naghshineh, P. Kermani, and J. Copeland, "Fair Media Access for Wireless LANs," in *IEEE GLOBECOM '99*, Rio de Janeiro, Brazil, Dec. 1999.

[10] D.-J. Deng and R.-S. Chang, "A Priority Scheme for IEEE 802.11 DCF Access Method," *IEICE Transactions on Communications*, vol. E82-B, no. 1, pp. 96–102, Jan. 1999.

[11] M. Barry, A. T. Campbell, and A. Veres, "Distributed Control Algorithms for Service Differentiation in Wireless Packet Networks," in *Proc. IEEE INFOCOM*, Anchorage, AK, USA, Apr. 2001.

[12] S. J. Golestani, "A Self-Clocked Fair Queuing Scheme for Broadband Applications," in *Proc. IEEE INFOCOM*, Toronto, Canada, June 1994, pp. 636–646.

[13] I. Aad and C. Castelluccia, "Differentiation Mechanisms for IEEE 802.11," in *Proc. IEEE INFOCOM*, Anchorage, AK, USA, Apr. 2001.

[14] R. Yavatkar, D. Hoffman, Y. Bernet, F. Baker, and M. Speer, "SBM (Subnet Bandwidth Manager): A Protocol for RSVP-based Admission Control over IEEE 802-style networks," May 2000, internet RFC 2814.

[15] A. Kochut, A. Vasan, A. U. Shankar, and A. Agrawala, "Sniffing out the Correct Physical Layer Capture Model in 802.11b," in *Proc. ICNP*, 2004.

[16] H.-L. Chao and W. Liao, "Fair Scheduling in Mobile Ad Hoc Networks with Channel Errors," *IEEE Transactions on Wireless Communications*, vol. 4, no. 3, pp. 1254–1263, May 2005.

[17] T. S. E. Ng, I. Stoica, and H. Zhang, "Packet Fair Queuing Algorithm for Wireless Networks with Location-dependent Errors," in *Proc. IEEE Infocom*, 1998.