Priority and Fair Scheduling in a Wireless LAN

Anurag Dugar, Nitin Vaidya, Paramvir Bahl*
adugar@ee.tamu.edu, vaidya@cs.tamu.edu, bahl@microsoft.com

Technical Report TR01-006
Computer Science Department
Texas A&M University
April 7, 2001[1]

Abstract

In recent years, fair scheduling and quality of service (QoS) in wireless LANs have received significant attention from the networking research community. This paper presents a distributed *algorithm for priority and fair scheduling in a wireless LAN. The proposed protocol is derived from HIPERLAN which is a wireless LAN standard defined by ETSI. The proposed protocol supports multiple priorities, as well as a mechanism (using weights) for controlling the way bandwidth is shared by flows within a given priority level.*

Keywords
Wireless LAN, Priority Scheduling, Weighted Fairness.

I. Introduction

FAIRNESS is an important issue when accessing a shared wireless channel. With *fair scheduling*, different flows wishing to share the wireless channel can be allocated bandwidth in proportion of their "weights". Also, with the increasing use of wireless LANs, there is a need for supporting quality of service and real time applications in the wireless LANs. The scheme proposed in this report supports multiple priority levels along with weighted fair sharing of a wireless channel.

Much research has been performed on "fair queueing" algorithms for achieving a *fair* allocation of bandwidth on a shared link (for e.g. [6], [9], [10], [12], [13], [15]). There has been some work on achieving fairness using distributed MAC protocols for wireless networks (for e.g. [7], [17], [18], [19]). Our recent related work [1] proposed a protocol to achieve weighted fairness where the nodes having different weights are allocated bandwidth in proportion to their weights. However, this protocol can exhibit short-term unfairness for some nodes when their transmissions collide.

There has also been work on distributed protocols that take *priorities* into account when performing medium access control and can support real-time applications [4], [8], [14]. Some interesting work on distributed scheduling algorithms for QoS support and real-time traffic on a wireless LAN has also been performed [3], [5], [16]. But none of these schemes supports multiple priorities along with weighted fairness.

*Paramvir Bahl is with Microsoft Research, Redmond, WA.
[1]Revised version of the Technical Report TR01-003, March 2001.

II. Proposed Scheme

The channel access mechanism in the proposed model is based on carrier sensing and a contention resolution scheme. The contention resolution scheme proposed in this paper borrows ideas from HIPERLAN [5] and [1]. Every backlogged node in the system senses the channel. If the channel is found to be idle for an "inter-round spacing" (*irs*), say, N slots, then the backlogged nodes initiate a contention resolution cycle. In order to ensure that a newly backlogged node does not enter an ongoing contention resolution cycle, we bound the maximum possible idle time during each cycle to be less than the *irs* duration (i.e., less than N slots).

In the proposed approach, every node in the wireless LAN is assigned a priority and a weight. Every backlogged node first chooses a backoff interval and then determines a tuple of the form (p, c, n, $d_{(n-1)}$,..., d_0) which is used for contention resolution. These are described in more detail in the following subsections.

A. Choosing a Backoff Interval

This scheme borrows the idea of choosing the backoff interval from [1]. The essential idea is to choose a backoff interval as a function of length of the packet to be transmitted, and weight of the flow to which the packet belongs, in order to achieve fair allocation of bandwidth. Specifically, a backlogged node i picks a backoff interval B_i for its k-th packet, P_i^k, as a function of its weight, ϕ_i, and packet length L_i^k, as follows :

$$B_i = \left\lfloor Scaling\_Factor * \frac{L_i^k}{\phi_i} \right\rfloor \tag{1}$$

An appropriate choice of the *Scaling_Factor* allows us to choose a suitable scale for the backoff interval. To reduce the possibility of collisions the B_i value chosen above is randomized as follows:

$$B_i = \lfloor \rho * B_i \rfloor \tag{2}$$

where ρ is a random variable uniformly distributed in the range [0.9,1.1].

B. Contention Resolution

Once a backlogged node has chosen a backoff interval, it constructs a tuple of the form (p,c,n,$d_{(n-1)}$,...,d_0). The elements of the tuple are assigned the following meaning:

- p: This is the most significant element in the tuple and denotes priority of the node. With base N, our approach can support up to N priority classes named P0, P1, etc.
- c: This element represents *collision status* of a node. Every node i maintains a collision counter $ccntr_i$. This counter is incremented on every collision suffered by a node and is reset to 0 after every successful transmission attempt by the node. If the collision counter for a node is greater than 0, then c is set to zero; else it is set to 1.
- n: This element denotes the number of digits in the backoff interval's representation in base N (underlying assumption is that n would not exceed $N-1$). We choose N as 10 for our simulations.
- d_i: denotes the i-th digit (0-th digit being the least significant) in the base N representation of the chosen backoff interval.

Whenever a backlogged node finds the channel to be idle for *irs* duration, it initiates a contention resolution cycle using its tuple. Recall that *irs* interval is chosen as the time duration for N slots. Each element in this tuple signifies a phase in the contention resolution cycle. Only those nodes which win the first phase, get to go onto the second phase and so on until the last phase, after which we will have potential winner(s) of the contention resolution cycle. During the i-th phase, the i-th digit, say t_i, in the tuple (p,c,n,$d_{(n-1)}$,...,d_0) is used as follows. In the i-th phase, a node listens to the channel for t_i slots. If it hears a transmission while it is listening, it drops out of contention. Otherwise, it transmits a *burst*[2] after listening for t_i slots and goes into the next phase. All the nodes which make a transition from i-th phase to the $(i+1)$-th phase have identical most significant i elements in their tuples.

C. Transmission and Recalculation of backoff intervals

The winner of the contention resolution cycle, node i, transmits a Request-to-Send (RTS) packet (similar to IEEE 802.11). If the receiver is willing to accept the data packet, it responds by sending a Clear-to-Send (CTS) packet on receipt of the RTS. When the sender gets the CTS, it sends the data packet and piggybacks its backoff interval B_i and its priority level on the data packet. A node j belonging to the same priority class as node i and which had contended with node i, recalculates its backoff interval on listening to the transmission as follows:

$$B_j = B_j - B_i \tag{3}$$

Node j then reconstructs the tuple on the basis of this new, smaller backoff interval. Now, when any node finds the channel idle for *irs* period, it initiates a new contention resolution cycle. This procedure is designed to ensure that there is a fair sharing of bandwidth among flows within each priority class.

D. Collision Resolution

Every node i maintains a collision counter, $ccntr_i$. This counter, $ccntr_i$, is incremented by 1 on every collision suffered by node i and is reset to zero after every successful transmission by node i. Now if two or more nodes belonging to the winning priority class had chosen the least backoff interval at the same time, they all would emerge as winners of the contention stage and would end up in collision. Since these colliding nodes were winners of the contention stage, they should get priority access over all other nodes within the same priority class. This is accomplished in the following manner: Suppose Node i sends a RTS and doesn't get back a CTS within the timeout duration. It then assumes that its RTS collided with a transmission from some other node (which indeed would be the case since we are assuming an error free channel). It then takes the following actions:

- Node i increments $ccntr_i$ by 1.
- Node i chooses a new B_i uniformly distributed in $[1, 2^{ccntr_i - 1} * CollisionWindow]$, where *CollisionWindow* is a constant parameter.
- Node i constructs a new tuple on the basis of this new B_i
- Node i waits for the channel to be idle for *irs* period and then initiates a contention resolution cycle using the new tuple.

[2]Burst is a control frame which is transmitted just to occupy the channel so that the other nodes listening to the channel can sense the channel as busy.

Let us assume that node i's priority class was the highest during the next contention resolution cycle. Now all the backlogged nodes within this class would check their *collision status*. Since node i has a non-zero collision counter, its *collision status* would be zero (as already explained while defining the elements of the tuple). Hence, node i would transmit a burst without waiting for any slot and then move onto the next element in the tuple. But for all other backlogged nodes within the same class who have not suffered a collision, *collision status* would be set to 1. Hence, all these nodes would have to wait for one slot before they can transmit a burst and move onto the next element in the tuple. While these nodes would be listening to the channel for one slot, they would hear the burst transmitted by node i and would subsequently drop out of contention. Hence node i would get access priority over all these nodes.

III. Performance optimization

The contention resolution mechanism in the proposed scheme has been slightly modified in order to reduce the overhead and optimize the performance. We do away with the collision status (second element in the tuple) and assign access priorities to the colliding nodes in a more efficient manner. As discussed earlier, every node i maintains a collision counter $ccntr_i$ which is incremented on every collision suffered by a node and is reset to 0 after every successful transmission attempt by the node. When the nodes reach the second phase of contention resolution (i.e. after priority resolution), they check their collision counter. If the collision counter for a node is greater than 0, then it transmits a burst and then moves on to the next element in the tuple, else it moves on to the next element in the tuple without transmitting a burst. Say node i had suffered a collision and was the highest priority node during the next contention resolution cycle. Since node i has a non-zero collision counter, it would transmit a burst and move on the next element in the tuple. But all other backlogged nodes within the same class would move on to the next element (n, i.e the number of digits in their backoff interval, which is guaranteed to be non-zero) without transmitting a burst. While these nodes would be listening to the channel for n slots, they would hear the burst transmitted by node i and would subsequently drop out of contention. Hence node i would get access priority over all these nodes.

IV. Simulation Model and Performance Evaluation

In this section, we present performance evaluation results for the proposed scheme. The scheme was implemented using ns-2 simulator [2]. The ns-2 simulator includes a module to simulate the Distributed Coordination Function (DCF) function in IEEE 802.11. We modified this module to simulate the proposed scheme. The channel bandwidth is taken to be 2 Mbps and the channel is assumed to be error free.

We compare the proposed scheme with the IEEE 802.11 standard [11] and the Distributed Fair Scheduling scheme proposed in [1] . In our simulation model, if we have a LAN with n nodes, we set up $n/2$ flows (n is always chosen to be an even number) – flow i is set up from node $2i$ to node $2i$+1 (the nodes are numbered 0 through $n-1$). The choice of the destination nodes for the flows is arbitrary, and any destination could have been chosen for each flow without affecting the results.

A. *Fair Scheduling*

The figures presented in subsection A consider the case where the $n/2$ flows (in case of a LAN with n nodes) have identical weights and priorities – the chosen weight for each flow is $2/n$ (this choice is arbitrary, and similar results hold for other choices too) and the priority of each node is P0. All the flows are generating traffic at the same rate and are always backlogged. Also, the aggregate demand by the flows exceeds the channel capacity.

A.1 Comparison of Proposed Scheme with IEEE 802.11 DCF

In the following figures we compare the aggregate throughput and the fairness index achieved by the proposed scheme and 802.11 as a function of the number of nodes on the wireless LAN . The scaling factor for the proposed scheme was chosen to be 0.02 in these simulations.

• Fairness Index

For environments where all flows are always backlogged, we evaluate a *fairness index* [20] as follows, where T_f denotes throughput of flow f, and ϕ_f denotes weight of flow f.

$$\text{Fairness index} = \frac{\left(\sum_f T_f/\phi_f\right)^2}{\text{number of flows} * \sum_f (T_f/\phi_f)^2}$$

As can be seen from Figure 1, the fairness index achieved by IEEE 802.11 degrades as the number of nodes on the wireless LAN is increased. On the other hand, the proposed scheme is able to achieve a consistently high fairness index (almost close to 1) even when the number of nodes scales up.

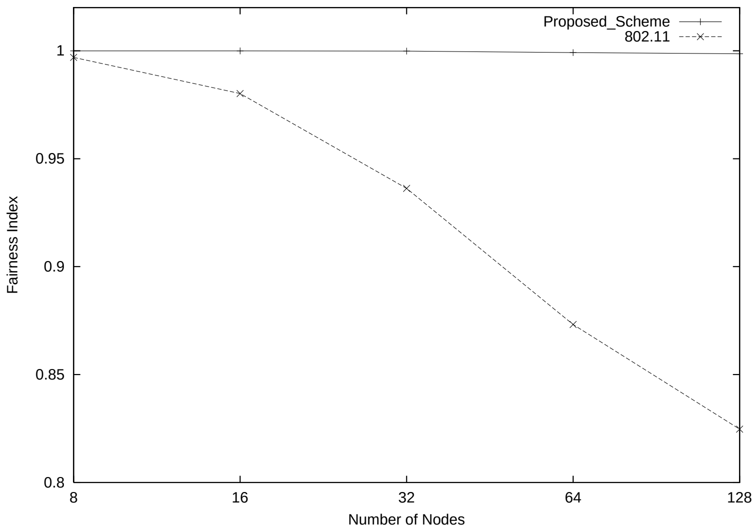

Fig. 1. A comparison of fairness index achieved by 802.11 and the proposed scheme.

• Aggregate Throughput

As can be seen from Figure 2, IEEE 802.11 tends to achieve a higher aggregate throughput – this illustrates the trade-off between throughput and fairness. The proposed scheme improves fairness, but possibly at the cost of lower aggregate throughput (due to longer durations of time spent on channel access protocol).

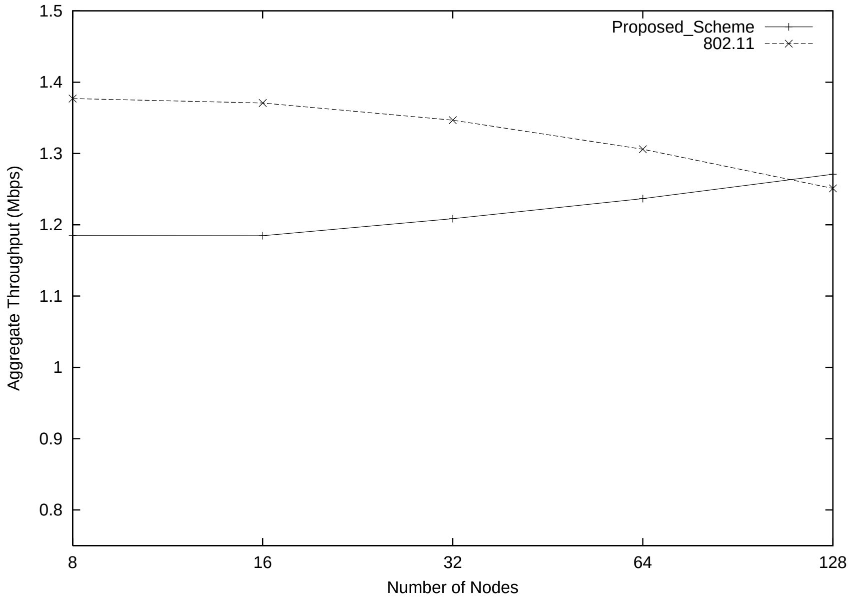

Fig. 2. A comparison of aggregate throughput achieved by 802.11 and the proposed scheme.

A.2 Comparison of the Proposed Scheme with DFS

We now compare aggregate throughput and fairness index achieved by the proposed scheme and DFS [1] as a function of the scaling factor. The wireless LAN had 32 nodes in these simulations.

- Fairness Index

In Figure 3, when the scaling factor is very small, the fairness index degrades in case of DFS. In DFS, when the scaling factor is very small, there is a high likelihood of collisions which can potentially exhibit unfairness towards the colliding nodes because of access priority reversals. On the other hand, in the proposed scheme, the colliding nodes within a class get access priority over other backlogged nodes within the same class. As a result, the proposed scheme achieves a high fairness index even for very small scaling factors. As for higher scaling factors, they result in an increase in fairness, and both DFS and the proposed scheme do well in such cases. This figure illustrates that unlike DFS, the proposed scheme is relatively insensitive to the choice of the scaling factor.

- Aggregate Throughput

In Figure 4, as the scaling factor is increased, the aggregate throughput achieved by DFS declines sharply. In DFS, large scaling factors result in large backoff intervals leading to a greater overhead and reduced aggregate throughput. On the other hand, in the proposed scheme, there is only a slight increase in the overhead with the scaling factor even though it results in large backoff intervals. This is because of the fact that our scheme reduces the time spent in counting down backoff interval by breaking it down into base N elements (N is chosen to be 10 in our simulations) and then using these elements during contention resolution cycle. Hence, as the scaling factor is increased, the aggreagate throughput achieved by the proposed scheme drops relatively slowly while there is an improvement in the fairness achieved.

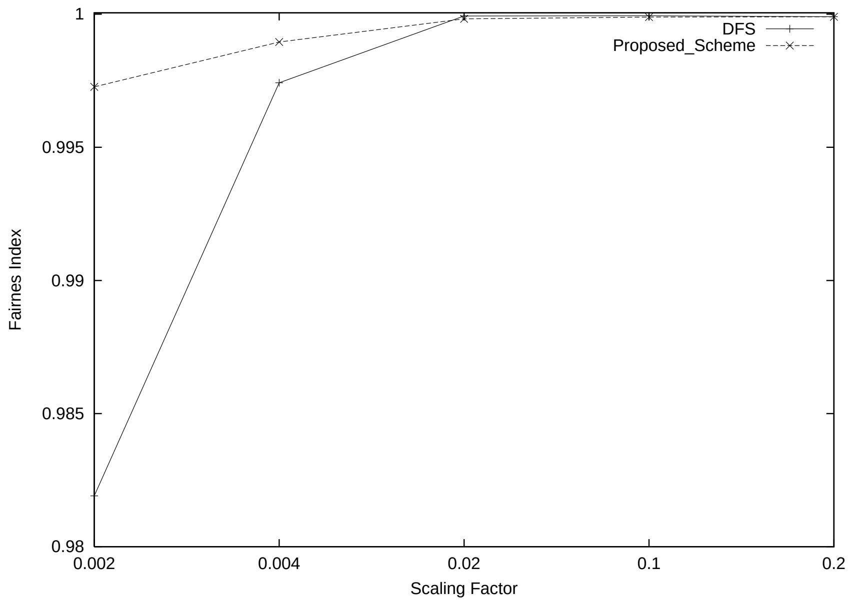

Fig. 3. A comparison of fairness index achieved by DFS and the proposed scheme

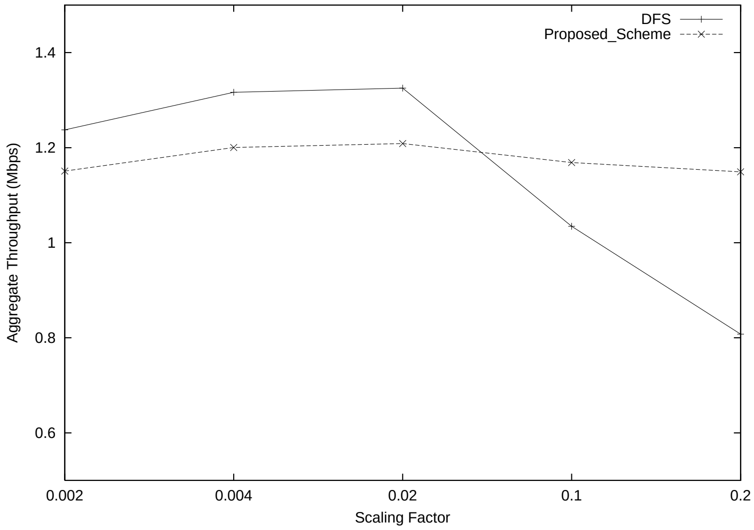

Fig. 4. A comparison of aggregate throughput achieved by DFS and the proposed scheme.

As can be seen from Figures 3 and 4, when the scaling factor is 0.02, DFS achieves better results than the proposed scheme. However, the performance of DFS is very sensitive to changes in the scaling factor and degrades as the value of the scaling factor deviates from 0.02. Whereas the proposed scheme achieves a fairly good performance over a wide range of scaling factors. When the number of nodes in the LAN is large, we would like to choose a large scaling factor to reduce the probability of collisions and maintain fairness. At the same time we would like to maximize the aggregate throughput achievable. Under such circumstances, the proposed scheme offers much more flexibilty as compared to DFS in choosing the scaling factor to achieve the desired objectives.

B. Priority Scheduling and Weighted Fairness

Let us consider the case where there are 20 nodes (i.e., 10 flows) in the wireless LAN. 5 of these flows (0,1 2,3 and 4) are of priority level P0 and the other 5 (5,6,7,8 and 9) are of priority level P1 (P0 being the higher priority). All the high priority flows have an equal weight of 1. The low priority flows have been assigned the weights 0.2, 0.4, 0.6, 0.8 and 1 respectively. The 5 high priority flows generate CBR traffic at the rate of 0.05Mbps, 0.1Mbps, 0.15 Mbps, 0.2Mbps and 0.25Mbps respectively. The rates are chosen so that the aggregate demand by the high priority flows is less than the channel capacity. The low priority flows, when present, generate CBR traffic at the rate of 0.5 Mbps each. The packet size is fixed and is identical for all flows in the simulations reported here. We present the results for the throughput achieved by the high priority flows with and without competing low priority flows. As can be seen from the Figure 5, since the aggregate demand by the high priority flows is below the channel capacity, each high priority flow gets a share of the bandwidth equal to its demand irrespective of its weight. The bandwidth leftover by the high priority flows is allocated to the low priority flows in proportion of their weights. Also, we can observe from Figures 5 and 6 that the demand of high priority flows is met whether or not there is competing low priority traffic.

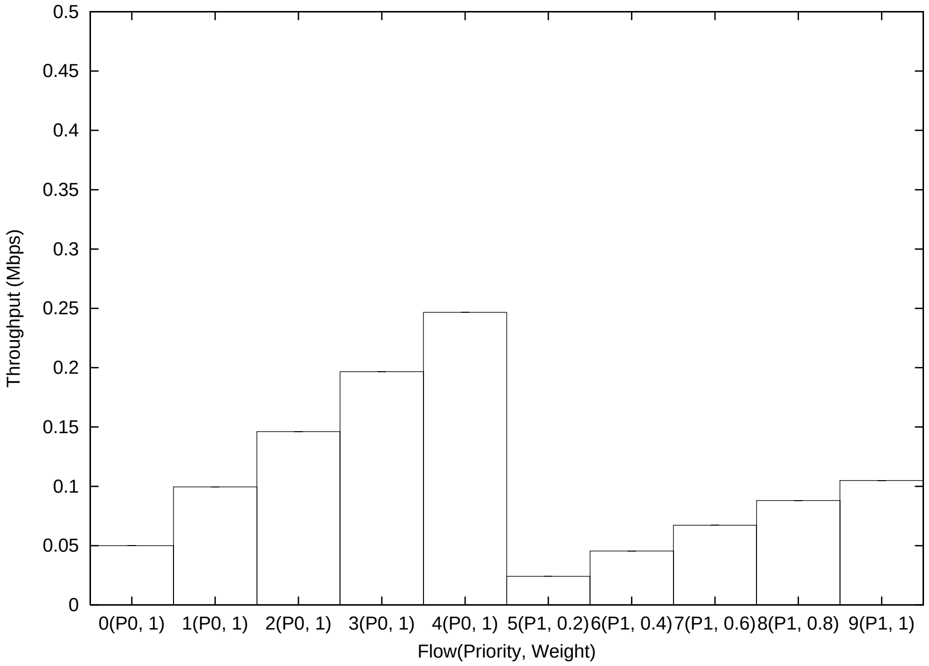

Fig. 5. Throughput achieved by the high priority flows with competing low priority traffic.

However, if the aggregate demand of the high priority flows exceeds the channel capacity then it is desirable that there be a weighted fair allocation of the bandwidth to these high priority flows. Let us consider the case when there are 10 nodes (i.e. 5 flows) in the wireless LAN. All the flows have the same priority, P0, but different weights. The weights of the flows are 1/2, 1/4, 1/6, 1/8, and 1/10 respectively. Each of these flows generates CBR traffic at the same rate and the aggregate demand exceeds the channel capacity. The packet size is fixed and is identical for all flows in the simulations reported here. We present the results

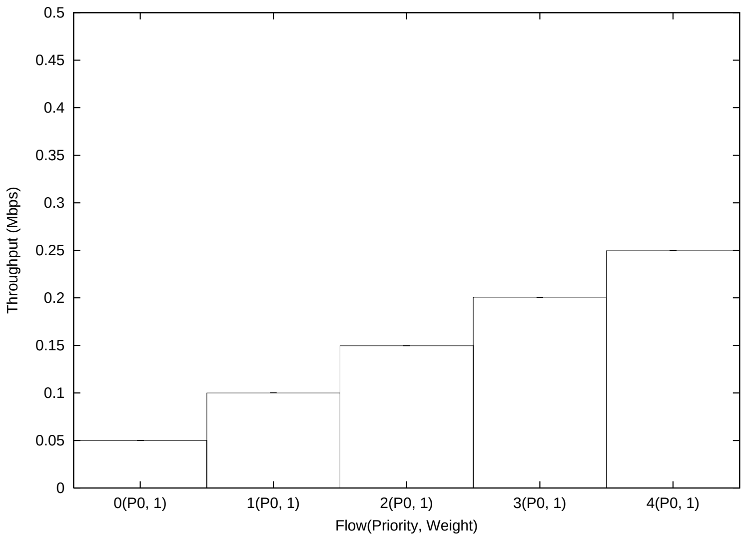

Fig. 6. Throughput achieved by the high priority flows with no competing low priority traffic.

for (throughput/weight) ratio achieved by each flow. As can be seen from Figure 7, the (throughput/weight) ratio remains more or less constant implying that each flow is getting a bandwidth allocation in proportion to its weight.

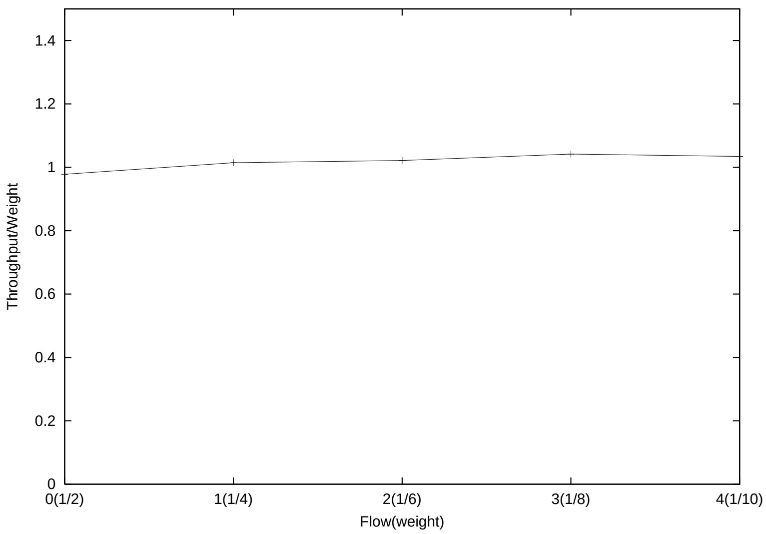

Fig. 7. Throughput/Weight achieved by flows having different weights.

V. Conclusions

This paper proposes a distributed medium access scheme for supporting multiple priorities and weighted fair scheduling in a wireless LAN, combining the techniques used in HIPERLAN [5] and DFS [1]. Performance results show that the proposed protocol can support multiple priorities and can allocate bandwidth to the flows within the same class in proportion of their weights. The ideas presented in this paper can be used to support QoS and real time applications in a wireless LAN. Also, these ideas may potentially be extended to multi-hop wireless networks, which is a subject of on-going research.

VI. Acknowledgements

This research was supported in part by Microsoft Research and National Science Foundation.

References

[1] N.H. Vaidya, P. Bahl, and S. Gupta, "Distributed Fair Scheduling in a Wireless LAN," Sixth Annual International Conference on Mobile Computing and Networking, Boston, August 2000.
[2] K. Fall and K. Varadhan, "*ns* Notes and documentation," tech. rep., VINT Project, UC-Berkeley and LBNL, 1997.
[3] J. Deng and R.S. Chang, "A Priority Scheme for IEEE 802.11 DCF Access Method," IEICE TRANS. COMMUN., VOL.E82-B, NO.1 January 1999.
[4] K. Ramamritham and W. Zhao, "Virtual time CSMA protocols for hard real-time communication," IEEE Transactions on Software Engineering,13(8):938–952, 1987.
[5] G. Anastasi, L. Lenzini, and E. Mingozzi, "Stability and Performance Analysis of HIPERLAN", Proc. of IEEE-Infocom, 1998.
[6] J. C. R. Bennett and H. Zhang, "Wf2q: Worst-case fair weighted fair queueing," in *INFOCOM'96*, March 1996.
[7] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, "MACAW: A media access protocol for wireless LANs," in *ACM SIGCOMM*, pp. 212–225, August 1994.
[8] G. L. Choudhury and S. S. Rappaport, "Priority access schemes using CSMA-CD," *IEEE Transactions on Communications*, vol. COM-33, pp. 620–626, July 1985.
[9] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," in *Proc. SIGCOMM*, September 1995.
[10] P. Goyal, H. M. Vin, and H. Cheng, "Start-time fair queueing: A scheduling algorithm for integrated services packet switching networks," *IEEE/ACM Transactions on Networking*, vol. 5, pp. 690–704, October 1997.
[11] IEEE, "IEEE std 802.11 - wireless LAN medium access control (MAC) and physical layer (PHY) specifications," 1997.
[12] S. Keshav, "On the efficient implementation of fair queueing," *Journal of Internetworking: Research and Experience*, vol. 2, pp. 57–73, September 1991.
[13] 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, June 1993.
[14] S. M. Sharrock and D. H. Du, "Efficient CSMA/CD-based protocols for multiple priority classes," *IEEE Transactions on Computers*, vol. 38, pp. 943–954, July 1989.
[15] M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round robin," in *SIGCOMM'95, Cambridge, MA, USA*, 1995.
[16] J. L. Sobrinho and A. S. Krishnakumar, "Real-time traffic over the IEEE 802.11 medium access control layer," *Bell Labs Technical Journal*, pp. 172–187, Autumn 1996.
[17] N. H. Vaidya and P. Bahl, "Fair scheduling in broadcast environments," Tech. Rep. MSR-TR-99-61, Microsoft Research, August 1999.
[18] H. Luo, S. Lu, and V. Bharghavan, "A New Model for Packet Scheduling in Multihop Wireless Networks," in *ACM MOBICOM*, August 2000.
[19] H. Luo and S. Lu, "A Self-Coordinating Approach to Distributed Fair Queueing in Ad Hoc Wireless Networks," to appear in *IEEE INFOCOM* , Anchorage, Alaska, Apr.26, 2001.
[20] R. Jain, G. Babic, B. Nagendra, and C. Lam, "Fairness, call establishment latency and other performance metrics," Tech. Rep. ATM_Forum/96-1173, ATM Forum Document, August 1996.