Distributed Weighted Fair Queuing in 802.11 Wireless LAN

Albert Banchs, Xavier Pérez
NEC Europe Ltd., Network Laboratories Heidelberg, Germany

***Abstract*— With Weighted Fair Queuing, the link's bandwidth is distributed among competing flows proportionally to their weights. In this paper we propose an extension of the DCF function of IEEE 802.11 to provide weighted fair queuing in Wireless LAN. Simulation results show that the proposed scheme is able to provide the desired bandwidth distribution independent of the flows aggressiveness and their willingness to transmit. Backwards compatibility is provided such that legacy IEEE 802.11 terminals receive a bandwidth corresponding to the default weight.**

***Index Terms*— Wireless LAN, IEEE 802.11, MAC, Weighted Fair Queuing, Bandwidth Allocation**

I. INTRODUCTION

Much research has been performed on "weighted fair queuing" algorithms for achieving the desired bandwidth allocation on a wired link [1], [2], [3], [4], [5], [6], [7]. With weighted fair queuing, the bandwidth received by a flow in a shared link is in proportion to the flow's weight.

Since Wireless LANs may be considered as just another technology in the communications path, it is desirable that the architecture for bandwidth allocation follows the same principles in the wireless network as in the wireline Internet, assuring compatibility among the wireless and the wireline parts.

The challenge in Wireless LAN is that we do not have all packets in a centralized queue, like in wired links, but we have them distributed in the wireless hosts. Therefore, we need to design a new MAC mechanism for Wireless LAN capable of providing the desired scheduling.

In this paper we propose a fully distributed approach, *Distributed Weighted Fair Queuing* (DWFQ), that extends the DCF mode of the 802.11 MAC protocol to distribute the bandwidth of the wireless network among the different flows proportionally to their weights.

The rest of the paper is structured as follows. Section II recalls the basics of the IEEE 802.11 standard. In Section III we describe the DWFQ architecture for providing weighted fair queuing in Wireless LAN. Simulation results are presented in the following section. The paper closes with an overview on related work and the conclusions (Sections V and VI).

II. THE IEEE 802.11 MAC LAYER

The basic IEEE 802.11 Medium Access mechanism is called Distributed Coordination Function (DCF) and is based on the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) protocol. In the DCF mode, a station must sense the medium before initiating the transmission of a packet. If the medium is sensed like being idle for a time interval greater than the Distributed Inter Frame Space (DIFS), then the station transmits the packet. Otherwise, the transmission is deferred and a backoff process is started.

Specifically, the station computes a random value in the range of 0 to the so-called Contention Window (CW). A backoff time interval is computed using this random value: $T_{backoff} = Rand(0, CW) * T_{slot}$, where T_{slot} is the slot time. This backoff interval is then used to initialize the backoff timer. This timer is decreased only when the medium is idle. The timer is frozen when another station is detected as transmitting. Each time the medium becomes idle for a period longer than DIFS, the backoff timer is periodically decremented, once every slot-time.

As soon as the backoff timer expires, the station accesses the medium. A collision occurs when two or more stations start transmission simultaneously in the same slot. An acknowledgment is used to notify the sending stations that the transmitted frame has been successfully received.

If an acknowledgment is not received, the station assumes that the transmitted frame was not received successfully and schedules a retransmission reentering the backoff process. To reduce the probability of collisions, after each unsuccessful transmission attempt, the CW is doubled until a predefined maximum (CW_{max}) is reached. After a (successful or unsuccessful) frame transmission, if the station still has frames queued for transmission, it must execute a new backoff process.

One problem concerning the carrier sensing of the DCF mode is the so-called hidden station problem. A station, that may not be within receiving range of a sending station and thus senses the medium idle, may however well be within sending range of the receiver of that ongoing communication and may thus cause a collision there, if it starts transmitting itself. To deal with the hidden station problem, the DCF MAC protocol can use the Request To Send (RTS) / Clear To Send (CTS) mechanism. When the RTS/CTS mechanism is applied, the winning station does not send data packets right away but sends a RTS packet to the receiving station, that responds with a CTS packet. If a station captures a RTS packet from another station and it is not the destination of the RTS packet it reads the intended transmission duration from the RTS packet and stays silent for that time. The same happens if only a CTS packet is received i.e. by a station outside the transmission range of the sender but within the range of the receiver. This guarantees that all stations within the range of either sender or receiver have knowledge of the transmission as well as of its duration. Besides reducing the collision probability in the case of a hidden station, another effect of the RTS/CTS mechanism is that it increases bandwidth efficiency since, if collisions occur, they do not occur with long data packets but with the relative small control packets.

The second access mechanism specified in the IEEE standard is built on top of DCF and it is called Point Coordination Function (PCF). It is a centralized mechanism, where one central coordinator polls stations and allows them undisturbed,

contention free access to the channel. In contention free access mechanism, collisions do not occur since the access to the channel is controlled by one entity.

In order to separate the different types of packets, different levels of access priority are implemented by defining 3 interframe spaces (IFS) of different length. They define the minimal time, that a station has to let pass after the end of a packet, before it may start transmitting a certain type of packet itself. After SIFS (Short IFS), the shortest interframe space, only acknowledgments, CTS and data frames in response to poll by the PCF may be sent. After PIFS (PCF-IFS), any frames from the contention free period may be sent in PCF-mode, after DIFS (DCF-IFS), the longest of the three interframe spaces, all frames in DCF-mode may be sent. This use of IFS allows the most important frames to be sent without additional delay and without having to compete for access with lower priority frames. It allows the prioritized access to the medium for the PCF mode over the contention mode frames in the DCF mode.

III. Distributed Weighted Fair Queuing

With weighted fair queuing, the bandwidth experienced by a flow i, r_i, is proportional to the *weight* that this flow has been assigned, W_i:

$$\frac{r_j}{W_j} = \frac{r_i}{W_i} \quad \forall i, \forall j \tag{1}$$

In the DCF approach, the bandwidth received by a flow depends on its CW: the smaller the CW, the higher the throughput. In DWFQ, weighted fair queuing in Wireless LAN is supported by the DCF function of the current standard with minor changes in the computation of the CW in order to give to each flow a bandwidth proportional to its *weight*.

According to the above explanation, terminals conforming to the IEEE 802.11 standard and DWFQ terminals compete with each other with different CW. In order to allow backward compatibility, the stations conforming to the IEEE 802.11 standard should behave as DWFQ stations with the default *weight*. In this paper we take the default *weight* equal to 1. This value corresponds to the basic service; *weights* smaller than 1 are not allowed, and any larger *weight* means "better than average" kind of treatment.

In the discussion and simulations of this paper, we will assume that all packets at a node belong to a single flow. The proposed approach, however, could be easily extended when multiple queues are maintained at each node, as discussed in Section III-C.

A. Contention Window Computation

The difficulty of DWFQ relies in determining the CW values that lead to the desired bandwidth distribution of Equation 1. The approach we have chosen for the calculation of the CW is a dynamic one.

In order to be able to properly adjust the CWs, we introduce a variable L_i, the *label*, defined as

$$L_i = \frac{r_i}{W_i} \tag{2}$$

where r_i is the estimated bandwidth experienced by flow i and W_i is its *weight*. The estimated throughput, r_i, is updated every time a new packet is transmitted:

$$r_i^{new} = (1 - e^{-t_i/K})\frac{l_i}{t_i} + e^{-t_i/K} r_i^{old} \tag{3}$$

where l_i and t_i are the length and inter-arrival time of the transmitted packet, and K is a constant. Following the rationale discussed in [7], in this paper we set $K = 100ms$.

With the above definition of the label L_i, the resource distribution expressed in Equation 1 can be achieved by imposing the condition that the label L_i should have the same value for all the flows:

$$L_i = L \quad \forall i \tag{4}$$

Note that the actual value of L can vary in time (depending on the number of flows for example).

Equation 4 is fulfilled by using the following algorithm: having calculated its own label L_i, each station includes its label in the header of the packets it sends. For each observed packet, if the L_i in the packet's header is smaller than the L_i of the station, the station increases its CW by a small amount, while in the opposite case it decreases its CW by a small amount. In this way, the L_i of all flows tend towards a common value, L.

The above explanation describes the basics of the algorithm. However, in the adjustment of the CW, there are additional aspects that have to be taken into account:

- We do not want the CW to increase above the values defined by the 802.11 standard; as argued above, for the backward compatibility reasons the basic service (with a *weight* equal to 1) uses the CWs defined in the 802.11 standard, and any higher *weight* should receive a "better than average" kind of treatment and therefore the values of the CW should be lower.
- If the low sending rate of the application is the reason for transmitting below the desired rate, then the CW should obviously not be decreased. This can be detected by the fact that in this situation the transmission queue is empty.
- CWs should not be allowed to decrease in such a way that they negatively influence the overall performance of the network. If the channel is detected to be below its optimum limit of throughput due to too small values for the CWs (i.e. overload), the CW should be increased. This aspect will be discussed in Section III-B.

The above considerations lead to the algorithm of Equation 5. This algorithm computes a value p which is used to scale the CW values defined in 802.11. Note that, besides this scaling of the CW, the backoff time computation algorithm is left as defined in the 802.11 standard (i.e. the Contention Window is doubled after every unsuccessful transmission attempt for a given number of times).

For each observed packet:

$$\begin{gathered}
\texttt{if } (L_{own} > L_{rcv}) \texttt{ then } p = (1 + \Delta_1)p \\
\texttt{else if } (queue_empty) \texttt{ then } p = (1 + \Delta_1)p \\
\texttt{else } p = (1 - \Delta_1)p \\
p = min\{p, 1\} \\
CW = p \cdot CW_{802.11}
\end{gathered} \tag{5}$$

where L_{own} is the label L_i calculated by the station, L_{rcv} is the label of the observed packet, and Δ_1 is computed as follows:

$$\Delta_1 = k\left|\frac{L_{own} - L_{rcv}}{L_{own} + L_{rcv}}\right| \quad (6)$$

where k is a constant equal to 0.01.

B. Overload

So far we have not discussed one important issue which is the *overload*. In fact, due to the nature of our protocol and in particular due to the dynamic way of adjustment of the size of the CW, a mechanism for controlling the overload is necessary.

As we can see in the algorithm of Equation 5, each station adjusts its CW only on the basis of its own requirements. Such "selfishness" can easily be disastrous, due to the following side effect of the small CWs. We have been arguing so far that, the smaller the CW for a given station, the larger the throughput received by this station. The other bad consequence of such a procedure is that the more stations have small CWs, the bigger the probability of a collision. One can easily see that, for a big number of stations with a high *weight*, this can lead to an absolute blockage of the channel. Once all of the stations start decreasing their CWs in order to get the desired relative throughput, the number of collisions will start increasing, leading to even smaller CWs, and as a consequence, continuous collisions. A solution to this problem is to extend the algorithm of Equation 5 with the following condition:

$$\begin{gathered}\text{For each observed packet:}\\ \texttt{if } (overload) \texttt{ then } p = (1+\Delta_2)p\\ \texttt{else if } (L_{own} > L_{rcv}) \texttt{ then } p = (1+\Delta_1)p\\ \texttt{else if } (queue\_empty) \texttt{ then } p = (1+\Delta_1)p\\ \texttt{else } p = (1-\Delta_1)p\\ p = min\{p, 1\}\\ CW = p \cdot CW_{802.11}\end{gathered} \quad (7)$$

where Δ_2 is a constant equal to 0.25.

Let us now explain how we actually detect *overload*. As we have mentioned before, a big number of stations trying to transmit with a high *weight*, i.e. decreasing their CWs, leads to an increase of the number of collisions. If we now provide each station with a collision counter[1], which determines how many collisions a packet experiences before it is successfully transmitted, we can write the following simple condition determining overload

$$\texttt{if } (av\_nr\_coll > c) \texttt{ then } overload = true \quad (8)$$

where c is a constant that has to be properly adjusted. If c is too low, flows with high *weights* will not be allowed to decrease their CWs sufficiently, and as a consequence they will not be able to achieve the desired bandwidth distribution. On the other hand, if c is too large, the number of collisions in the channel will be very high and the overall performance will be harmed. This constant, therefore, represents a tradeoff between the level of accurateness of the bandwidth distribution and the efficiency (i.e. total throughput) of the channel. This tradeoff has been studied via simulation (see Section IV-D), and an optimum value for c has been chosen according to simulation results.

[1] Note that in 802.11 collisions can only be detected through the lack of the Ack. However, a missing Ack can also be caused by other reasons different than a collision. In the simulations section we study the impact into our algorithm of having missing Acks due to errors in the channel (see Section IV-G).

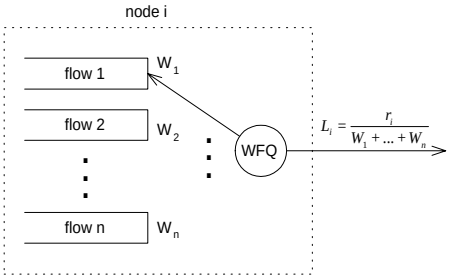

Fig. 1. Multiple Flows per Node.

The average number of collisions, (av_nr_coll), in Equation 8 is calculated after each successful transmission in the following way

$$av\_nr\_coll = (1-t) \cdot num\_coll + t \cdot av\_nr\_coll \quad (9)$$

where in order to smoothen its behavior, we use some sort of memory, taking into account the last calculated value of av_nr_coll (on the rhs of Equation 9). The constant t is a small number (in our case $t = 0.25$) playing the role of a smoothening factor.

C. Multiple Flows per Node

In our discussion of DWFQ we assumed that only one flow exists at each node. In general, it is possible that each node may maintain multiple flows locally. In this case we modify the DWFQ protocol as described below (see Figure 1).

- A node i transmitting n flows with weights $W_1, \ldots, W_n$ uses the label
$$L_i = \frac{r_i}{\sum_{j=1}^{n} W_j} \quad (10)$$
where r_i is the estimated aggregated bandwidth experienced by node. This gives to the node the total bandwidth necessary for all its flows.
- Node i uses a weighted fair queuing scheduler with weights $W_1, \ldots, W_n$ to choose the next packet to transmit. This distributes the total bandwidth of the node among its flows proportionally to their weights $W_1, \ldots, W_n$.

IV. Simulations

To test the performance of the architecture presented in this paper, we simulated it on a network consisting of a number of wireless terminals in a 2 Mbps Wireless LAN communicating with a fixed node through the Access Point (AP). These simulations were performed in ns-2 [8]. For this purpose, the algorithm of Equation 7 was inserted into the existing implementation of the 802.11 MAC DCF protocol in ns-2.

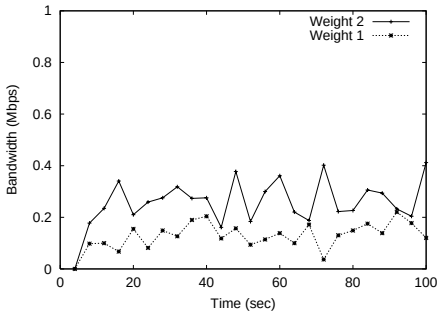

Fig. 2. Instantaneous Bandwidth Distribution.

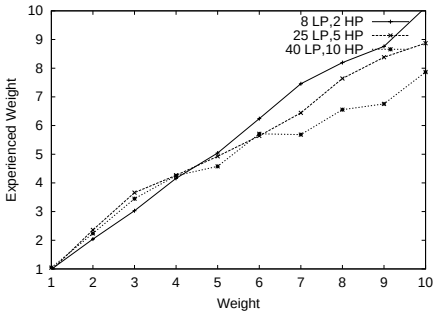

Fig. 3. Bandwidth Distribution as a function of the *weight*.

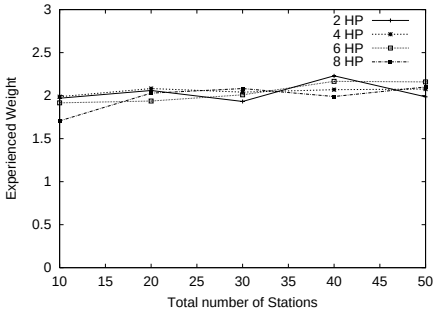

Fig. 4. Bandwidth Distribution as a function of the number of stations.

We chose to use the RTS/CTS mechanism in all cases. This mechanism, optional in the 802.11 standard, increases bandwidth efficiency in case of many collisions. Since our architecture may lead to larger number of collisions than the normal 802.11 MAC DCF, this mechanism can be especially beneficial in our case. Note, however, that the proposed DWFQ extension would also work without the RTS/CTS mechanism.

As already mentioned in Section III, we assume that all nodes are sending just one flow, except for the downlink in experiment IV-I.

A. Instantaneous Bandwidth Distribution

In DWFQ the desired bandwidth distribution is achieved by adjusting adaptively the CW of DWFQ stations according to the measured performance. Figure 2 shows this dynamic adjustment; the simulation corresponds to a scenario with a total number of 10 stations, 2 of them with a *weight* of 2 (high priority) and the rest with a *weight* of 1 (low priority). All stations are sending UDP CBR traffic with a packet size of 1000 bytes. It can be seen when comparing the instantaneous bandwidth of high priority and low priority stations that their ratio oscillates around the desired value.

B. Bandwidth Distribution as a function of the weight

With the proposed DWFQ extension, the throughput experienced by a station should be proportional to the *weight* assigned to its flow. Figure 3 shows the ratio between the throughput experienced by high priority (HP) and low priority (LP) stations (*experienced weight*) as a function of the *weight* assigned to the high priority stations when low priority stations have a *weight* equal to 1. Ideally, the resulting function should be the identity (i.e. a diagonal); that is, an *experienced weight* equal to the *weight*. In the figure it can be seen that the simulated results are quite close to the ideal. Only in the case of large weights and a large number of stations, the results obtained differ noticeably from the ideal case; however, not even in this case differences are too big (e.g. with 50 stations and a *weight* of 10, the *experienced weight* is 8).

C. Impact of the number of stations

The proposed algorithm for DWFQ relies on the experienced throughput estimated by each station. Note that the higher the number of stations, the lower the throughput received by each station. Since a low throughput is more difficult to estimate with exactitude than a high throughput, a large number of stations may negatively impact the performance of the algorithm. Figure 4 shows this impact when high priority stations have a *weight* of 2 and low priority ones have a *weight* of 1. Note that, in all cases, the experienced ratio between throughput of high and low priority stations keeps close to the desired value, which is 2. We conclude that the number of stations has a negligible impact on the *experienced weight*.

D. Impact of the parameter c

In Section III-B the constant c has been defined as the maximum average number of collisions allowed. This limit is needed in order to avoid loss of efficiency due to too small CWs.

Since we are using the RTS/CTS mechanism, the number of collisions will never be bigger than 8 (according to the standard, a packet is dropped after 8 RTS tries). Therefore, the chosen value for c must be in the range of $0 < c < 8$.

In order to analyze the impact of c we chose to use a scenario with a large number of stations (100 stations), half of them with very high weights ($weight = 6$). This scenario leads to many stations with very small CW, and, therefore a high number of collisions, in such a way that collisions are controlled by the parameter c. Note that in a scenario without many collisions the impact of c would be almost null.

Figures 5, 6 and 7 show the total throughput, the number of drops per successful packet and the *experienced weight* as a function of c in the above scenario. In these figures it can be seen that if the value of c is too high, the total throughput experienced is very low, and the percentage of losses very high. In the extreme case ($c > 7$) the throughput drops to 0 and the drops increase drastically. The reason for this is that with such values

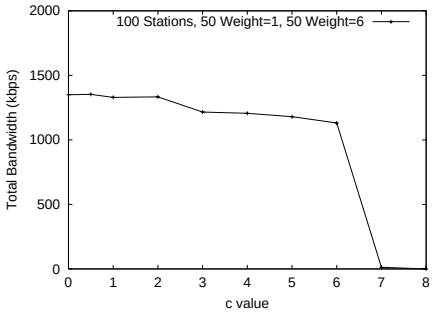

Fig. 5. Throughput as a function of *c*.

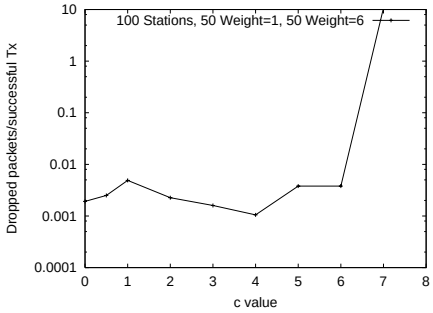

Fig. 6. Drops as a function of *c*.

of c, CWs are allowed to decrease too much and the probability of collision gets too big. Note that in this case low priority stations totally starve and the differentiation tends to infinite, since the whole bandwidth is used by high priority stations.

On the other hand, if the value of c is too low, we obtain a good total throughput and very low losses, but we do not achieve the desired differentiation. In the limit ($c = 0$) there is no differentiation at all and high priority stations get exactly the same throughput as low priority ones (i.e. *experienced weight* $= 1$). The reason for this is that, with such values of c, CWs are not allowed to decrease below the values for $weight = 1$ (i.e. the ones defined in the 802.11 standard), and, therefore, the DWFQ extension defined in this paper is deactivated.

As a conclusion, c expresses a tradeoff between efficiency and differentiation, and it can be adjusted via administration depending on specific user preferences. In this paper we have chosen to use an intermediate value: $c = 5$. With this value of c, a good level of differentiation is achieved while conserving a good overall efficiency.

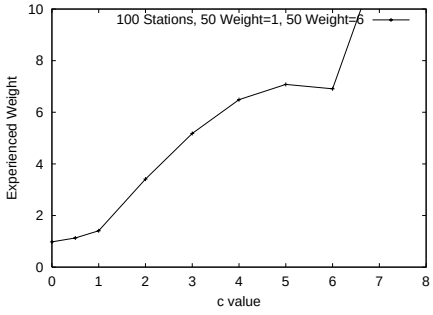

Fig. 7. Experienced weight as a function of *c*.

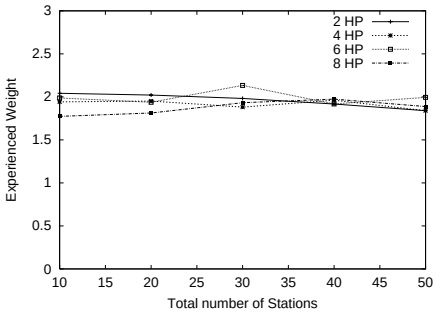

Fig. 8. Impact of 802.11 terminals.

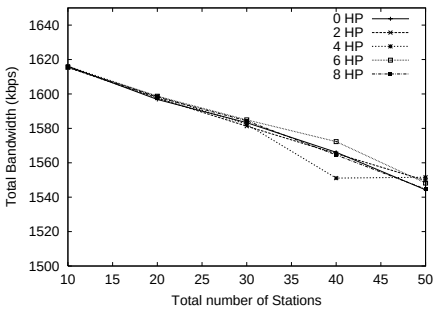

Fig. 9. Channel utilization.

E. Backwards Compatibility

As explained in Section III, stations conforming to the 802.11 standard receive in our architecture the same treatment as the stations with a *weight* equal to 1. Legacy 802.11 stations, however, do not carry in the header the L_i field, and this can have an impact into the overall performance of the DWFQ extension. This impact is studied in the simulation results shown in Figure 8. In this simulation, the number of stations with the DWFQ extension implemented is kept to 10, and the rest of the stations are legacy 802.11 terminals. Figure 8 shows the ratio between the throughput of high priority stations (*weight* 2) and low priority stations (*weight* 1) as the number of 802.11 terminals increases. It can be seen that this ratio is very close to the desired value, independent of the number of 802.11 terminals.

F. Channel utilization

Having DWFQ stations with a CW smaller than the CW defined in the current standard can impact the channel utilization. Figure 9 shows the channel utilization in the same scenario than the described for experiment IV-C, and compares it to the channel utilization with the current standard. It can be seen that the channel utilization keeps always close to the channel utilization of the current standard.

G. Channel Errors

Considering a non-ideal channel, a not received ACK can be due to a channel error. As discussed in Section III-B we have introduced a collision counter which counts as a collision every sent packet (RTS) for which an ACK (CTS) has not been received. The effect of the channel errors in the collision counter would be the interpretation of a channel error as a collision.

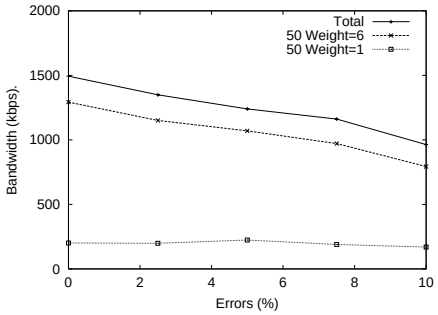

Fig. 10. Level of differentiation as a function of the error rate.

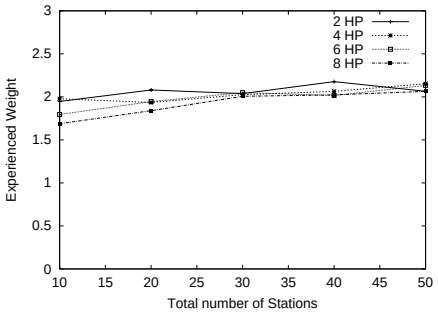

Fig. 12. Sources UDP ON/OFF 500 ms.

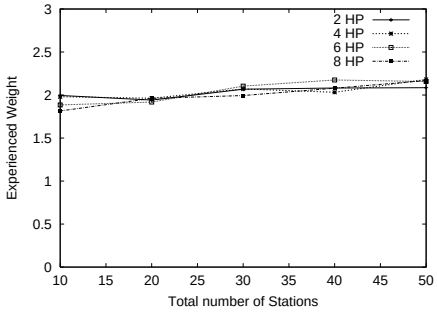

Fig. 11. Sources UDP ON/OFF 1 ms.

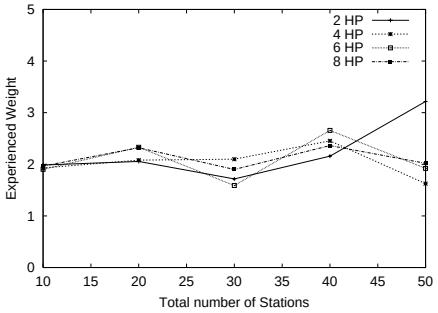

Fig. 13. TCP Sources.

This could lead to assume falsely overload due to channel errors. The result would be an unnecessary increase of the CW, leading to a lower level of differentiation.

We have studied this impact under the extreme scenario of Section IV-D with a value of c equal to 5. We can observe from Figure 10 that the level of differentiation (*experienced weight*) is affected by the percentage of errors, as expected. However, even in this extreme scenario and with a high error percentage (10%), we still keep a reasonably high level of differentiation.

H. Impact of bursty traffic

The simulations shown so far correspond to a constant traffic (UDP CBR sources). In order to gain a better understanding of the impact of different traffic sources to the performance of DWFQ, we have simulated it under bursty traffic (UDP ON/OFF sources).

In order to show the impact of different burst sizes, we performed two different simulations: one with a small burst (ON/OFF periods of 1 ms in average), and one with large bursts (ON/OFF periods of 500 ms in average). The simulation scenario was the same as the described in experiment IV-C.

Figure 11 shows the results when the ON/OFF periods are of 1 ms. Note that these results are very similar to the results of Figure 4 (CBR traffic), which means that short ON/OFF periods do not impact the performance of a station. In Figure 12 it can be seen that the results for large ON/OFF periods are also very similar to the results of Figure 4, with a slightly higher oscillation.

I. TCP sources

Figure 13 shows the *weight* experienced by high priority stations for the scenario of experiment IV-C with TCP sources. It can be seen that there are quite high oscillations in the *experienced weight* obtained, specially when the number of stations is high. This oscillation is due to the congestion control algorithm used in TCP. However, in average, the results obtained tend to the desired ones.

Note that, in contrast to the previous experiments, in this case we have downlink traffic consisting of TCP acknowledgments. Since this traffic consists of several flows, we are in the multiple flows per node case. We used the solution explained in Section III-C to handle this case. We assigned to the flow i of TCP acknowledgments the same *weight* as the corresponding flow of TCP data packets. This is necessary in order to achieve the desired bandwidth distribution, since the TCP acknowledgments also impact the throughput of a TCP connection through the congestion control of TCP.

J. TCP vs UDP

When TCP and UDP flows compete with each other, the bandwidth distribution tends to favor UDP. This is because, in case of congestion, TCP backs off because of its congestion control mechanism, and UDP, without any kind of congestion control and therefore more aggressive, consumes the bandwidth left by TCP. An architecture for bandwidth allocation should overcome this different level of aggressiveness of the sources and provide all sources with their fair share of bandwidth independent of the congestion control algorithm they use.

To study the level of fairness between TCP and UDP achieved by DWFQ, we performed the following experiment: two high priority stations had a *weight* of 2, one sending an endless TCP flow and the other a UDP CBR flow. The remaining 8 stations had a *weight* of 1 (low priority) and were all sending

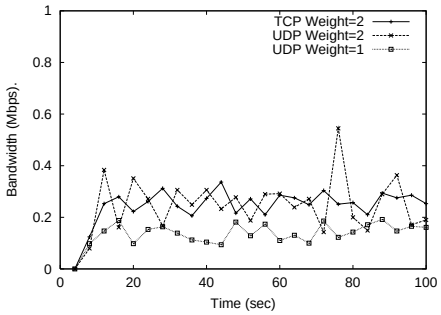

Fig. 14. TCP vs UDP

UDP CBR traffic. Figure 14 shows the instantaneous bandwidth achieved by the TCP and UDP high priority sources and one UDP low priority source. It can be seen that, in average, the resulting bandwidth distribution is the desired.

From this experiment we conclude that DWFQ provides TCP with a fair treatment with respect to UDP. This is because the DWFQ algorithm adapts the CW to the aggressiveness of the source: a less aggressive source, like TCP, will see its CW reduced until it receives the desired relative throughput, while a more aggressive source, like UDP, will achieve its desired relative throughput with a larger CW.

V. Related work

One possible approach for providing weighted fair queuing in Wireless LAN is to rely on centralized control and polling of backlogged wireless hosts [9], [10]. In contrast to these proposals, the architecture we propose is based on distributed control. We argue that distributed control results in more productive use of radio resources.

[11], [12], [13] and [14] are other proposals for bandwidth distribution relying on distributed control. These architectures are based on the idea of modifying the backoff time computation of the 802.11 standard to provide the desired bandwidth allocation, which is also the basis of our approach.

[11] and [12] propose the use of different CWs and different backoff increase parameters, respectively, without modifying the CW computation algorithm of 802.11. [13] proposes the use of different CWs and modifies the CW computation algorithm. The fact that the parameters in [11], [12] and [13] are static makes the resulting bandwidth distribution uncertain, as opposed to our proposal, in which the desired bandwidth allocation is achieved by modifying dynamically the CWs considering both the aggressiveness of the sources and their willingness to transmit.

The idea of modifying dynamically the backoff computation parameters had already been mentioned in [13]. However, [13] provides neither an algorithm nor simulation results for this dynamic adaptation.

[14] provides relative priorities for delay and throughput in a multi-hop wireless network. This approach piggybacks scheduling information onto RTS/DATA packets and then uses this information to maintain a scheduling table in each node. This table is then used to modify the computation of the backoff times. One major drawback of [14] as compared to our approach is its complexity. Moreover [14] does not provide backwards compatibility.

VI. Conclusions

In this paper we have proposed the DWFQ architecture for providing weighted fair queuing in wireless LAN. DWFQ provides a flow with an average bandwidth proportional to its *weight*, but does not give any guarantees for individual packets (i.e. DWFQ can exhibit short-term unfairness).

The design goals of DWFQ have been to keep the MAC protocol fully distributed, to minimize the migration effort from the current standard, and to provide backwards compatibility. We argue that a fully distributed MAC protocol is more efficient and flexible than a centralized one. We believe that the fact that DWFQ only requires minor changes in the computation of the CW facilitates the migration from the 802.11 standard. Finally, the computation of the CW has been designed in such a way that legacy 802.11 terminals receive the basic service in the proposed architecture.

The simulations performed show that DWFQ provides the desired bandwidth distribution among flows in a wide variety of scenarios. Because of the dynamic adaptation of the CW, this bandwidth distribution is independent of the level of aggressiveness of the sources and their willingness to transmit. Furthermore, simulation results show that DWFQ avoids harming channel utilization in case of overload.

References

[1] J. C. R. Bennet and H. Zhang, "Wf2q: Worst-case Fair Weighted Fair Queuing," in *Proceedings of IEEE INFOCOM*, San Francisco, CA, March 1996.

[2] A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queuing Algorithm," in *Proceedings of ACM SIGCOMM*, Cambridge, MA, September 1995.

[3] P. Goyal, H. M. Vin, and H. Cheng, "Start-time Fair Queuing: a Scheduling Algorithm for Integrated Serivces Switched Networks," *IEEE/ACM Transactions on Networking*, vol. 5, pp. 690–704, October 1997.

[4] S. Keshav, "On the Efficient Implementation of Fair Queuing," *Journal of internetworking: Research and Experience*, vol. 2, pp. 57–73, September 1991.

[5] 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.

[6] M. Shreedhar and G. Varghese, "Efficient Fair Queuing using Deficit Round Robin," in *Proceedings of ACM SIGCOMM*, Cambridge, MA, September 1995.

[7] I. Stoica, S. Shenker, and H. Zhang, "Core-Stateless Fair Queueing: Achieving Approximately Fair Bandwidth Allocations in High Speed Networks," in *Proceedings of ACM SIGCOMM '98*, Vancouver, Canada, August 1998, pp. 118–130.

[8] "Network Simulator (ns), version 2," `http://www-mash.cs.berkeley.edu/ns`.

[9] T. Nandagopal, S. Lu, and V. Bharghavan, "A Unified Architecture for the Design and Evaluation of Wireless Fair Queuing Algorithms," in *Proceedings of ACM MOBICOM*, Seattle, WA, August 1999.

[10] S. Lu, V. Bharghavan, and R. Srikant, "Fair Scheduling in Wireless Packet Networks," in *Proceedings of ACM SIGCOMM*, Cannes, France, August 1997.

[11] A. Ayyagari, Y. Bernet, and T. Moore, "IEEE 802.11 Quality of Service - White Paper," IEEE 802.11-00/028.

[12] A. Imad and C. Castelluccia, "Differentiation Mechanisms for IEEE 802.11," in *Proceedings of INFOCOM*, Anchorage, Alaska, April 2001.

[13] N. H. Vaidya, P. Bahl, and S. Gupta, "Distributed Fair Scheduling in Wireless LAN," in *Proceeding of MOBICOM*, Boston, MA, August 2000.

[14] V. Kanodia, C. Li, B. Sadeghi, A. Sabharwal, and E. Knightly, "Distributed Multi-Hop with Delay and Throughput Constraints," in *Proceeding of MOBICOM*, Rome,Italy, July 2001.