JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 23, 1707-1721 (2007)

Enhanced Fair Scheduling for IEEE 802.11e Wireless LANs*

KUN-NAN TSENG, KUOCHEN WANG+ AND HUNG-CHENG SHIH
Department of Computer Science
National Chiao Tung University
Hsinchu, 300 Taiwan
+*E-mail: kwang@cs.nctu.edu.tw*

As wireless LANs are gaining popularity, the demand for supporting multimedia and QoS-sensitive applications becomes more important than before. Although enhancements to the legacy IEEE 802.11 MAC to support QoS mechanisms have been proposed, they suffer from unfair allocation of bandwidth between high and low priority traffic. We propose a distributed enhanced fair scheduling (EFS) scheme that can conquer the above problem. With a fast backoff mechanism in the backoff timer decrement state and by dynamically adjusting backoff intervals according to the network load, we can enhance the performance of the EFS. We have evaluated the performance of the EFS through simulation. Experimental results show that the proposed EFS has better throughput performance than DFS by 13%, lower average MAC delay than DFS by 6% and the two have nearly equal fairness. Although the enhanced distributed channel access (EDCA) in IEEE 802.11e has better throughput and delay performance than EFS and DFS, it has very poor fairness. The contention free burst (CFB) mechanism in EDCA is the main factor that results in good throughput performance, lower average MAC delay and poor fairness. Our EFS is very suitable for applications that need strict fair bandwidth allocation, such as pay services.

Keywords: backoff interval, fair scheduling, IEEE 802.11e, quality of service, wireless LAN

1. INTRODUCTION

The IEEE 802.11 Wireless LAN is gaining a lot of popularity in recent years because of cost-effectiveness in building a wireless broadband network environment. Mobile computing devices such as portable computers and personal digital assistants become indispensable in our daily activities. With the increasing use of wireless LANs, there is high demand for supporting multimedia and QoS-sensitive applications. Because the IEEE 802.11 legacy MAC is only suitable for QoS-insensitive applications, the enhancements to the legacy MAC to support QoS mechanisms [1-4] were proposed. The enhancements provide service differentiation by statically assigning priorities to classes. A high priority class is assigned a shorter interframe space (IFS) and a shorter contention window (CW) than a low priority class. However, these approaches suffer from unfair allocation of bandwidth between high and low priority traffic. Various Fair scheduling schemes [5-9] were designed to overcome the above problem by allocating the bandwidth fairly between different traffic classes. In this paper, we propose an *enhanced fair*

Received September 6, 2005; accepted February 6, 2006.
Communicated by Yu-Chee Tseng.
* This work was supported by the NCTU EECS-MediaTek Research Center under grant Q583 and National Science Council under grant NSC93-2213-E-009-124. A brief version of this paper was presented at *the 11th Mobile Computing Workshop*, Taoyuan, Taiwan, March 2005: Sponsor – National Science Council (NSC).

scheduling (EFS) scheme that can provide better throughput and delay performance than the *distributed fair scheduling* (DFS) [10] and maintain nearly equal fairness.

Many researches on fair scheduling algorithms for achieving fair allocation of bandwidth on a shared wired link [5-7, 11] have been proposed. All fair scheduling algorithms are based on the fluid fair scheduling model [11]. In the model, packet flows are modeled as fluid flows. Fluid fair scheduling guarantees that for an arbitrary time interval $[t_1, t_2]$, any two backlogged flows i and j are served in proportion to their weights, which are represented by the following equation:

$$\frac{W_i(t_1, t_2)}{\phi_i} = \frac{W_j(t_1, t_2)}{\phi_j} \tag{1}$$

where $W_i(t_1, t_2)$ and ϕ_i are the service amount received (bits) by flow i during time interval $[t_1, t_2]$ and the weight of flow i, respectively. However, in the real network world, systems handle flows at the granularity of packets rather than bits. Therefore, the main objective of packet fair queuing (PFQ) algorithm is to approximate the *Generalized Processor Sharing* (GPS) [11] as closely as possible. The GPS is an ideal fluid fair scheduling model. The most famous PFQ algorithm is the *weighted fair queuing* (WFQ) [12], equivalently a *packetized generalized processor sharing* (PGPS). However, WFQ has its drawbacks that it is not easy to implement because the cost of maintaining a priority sorting queue and the overhead of computing the virtual system time is very high. *Self clocked fair queuing* [6] and *start time fair queuing* [7] reduced implementation complexity of WFQ. The PFQ algorithms developed for wired networks cannot be directly applied to wireless networks because of bursty and location-dependent errors in wireless channels. In the next section, we will describe classical fair scheduling algorithms that emulate the PFQ algorithm in wireless networks.

The subsequent sections of this paper are organized as follows. In section 2, we describe previous work in this research area. We describe the proposed EFS in detail in section 3. In section 4, we evaluate the throughput and delay performance, and the fairness index of the proposed EFS via simulation. In section 5, we summarize our work.

2. RELATED WORK

We will review the enhanced distributed channel access (EDCA) [13, 14], and distributed fair scheduling (DFS) [10] in this section.

2.1 The Enhanced Distributed Channel Access (EDCA)

The fundamental MAC protocol [15] used in IEEE 802.11 can not support QoS-sensitive applications. Therefore, the IEEE 802.11e introduces the enhanced distributed channel access (EDCA) and hybrid coordination function (HCF) to support QoS [13]. Service differentiation is achieved through the introduction of access categories (ACs). Each AC on a station contends for a transmission opportunity (TXOP) [13] and has its own transmission queue. Each transmission queue has a different interframe space (called arbitrary interframe space – *AIFS*[*AC*]), and a different set of contention window

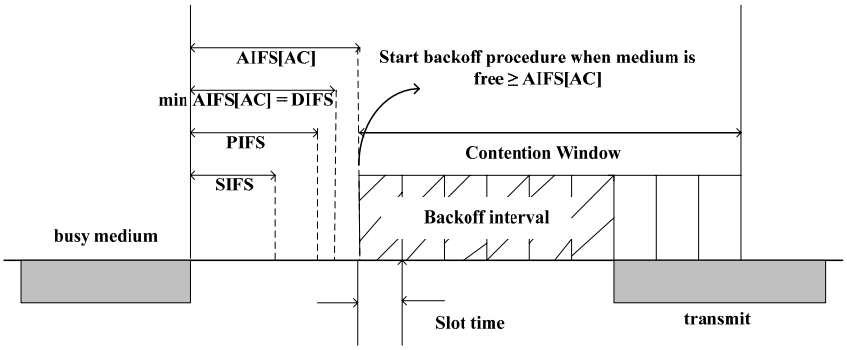

Fig. 1. IEEE 802.11e EDCA IFS relationships [13].

limits (*CWmin*[*AC*] and *CWmax*[*AC*]). Fig. 1 illustrates the EDCA IFS relationships [13]. Each AC starts its backoff procedure when the medium is idle for *AIFS*[*AC*] time. When an AC starts its backoff procedure, it chooses a backoff interval uniformly distributed in [0, *CWmin*[*AC*]]. The AC will transmit its packet immediately when the backoff interval counts down to zero. Although the IEEE 802.11e supports better QoS than the legacy IEEE 802.11, it suffers from unfair allocation of bandwidth between high and low priority traffic and has high throughput variability. Randomness in accessing the medium is the main factor resulting in high throughput variability.

2.2 The Distributed Fair Scheduling (DFS)

Some existing schemes [5, 10] using fair scheduling were designed to overcome the above unfair problem. This kind of schemes can provide relative differentiation, for example, specifying that one type of traffic should get twice as much bandwidth as some other type of traffic. For instance, Vaidya *et al*. [10] proposed Distributed Fair Scheduling (DFS), attempting to emulate Self-Clocked Fair Queuing (SCFQ) [6]. SCFQ is a centralized algorithm for packet scheduling on a link shared by multiple flows. In the SCFQ algorithm, the start and finish tags are calculated when a packet arrives in a flow. Alternatively, the start tag can be calculated when a packet reaches the front of its flow.

Like SCFQ, DFS determines the packet transmission based on the finish tag of each packet and the virtual time is updated in the same way as that in SCFQ. In the DFS, a packet with the smallest ratio between its length and weight receives the highest priority to transmit. The main idea of this scheme is to pick a backoff interval proportional to the finish tag of the packet. It attempted to emulate SCFQ in a distributed manner so as to transmit the packet with the minimum finish tag first. A backlogged node *i* picks a backoff interval B_i as a function of its weight, ϕ_i, and packet length L_i, as follows:

$$B_i = \lfloor \lfloor Scaling\_Factor * L_i / \phi_i \rfloor * \rho \rfloor \tag{2}$$

where ρ is a random variable uniformly distributed in [0.9, 1.1] and is introduced to reduce the possibility of collisions, and *Scaling_Factor* is a factor allowed us to choose a suitable scale for the backoff interval B_i [10].

When a collision occurs, a new backoff interval is calculated using the backoff procedure of the IEEE 802.11 standard [15] where the initial contention window is set to 4 [10]. The reason for choosing such a short contention window although a collision has occurred is that DFS intends to maintain fairness among nodes, and thus colliding stations should be able to send packets as soon as possible. However, mapping the QoS requirement to the weight is complicated.

3. DESIGN APPROACH: ENHANCED FAIR SCHEDULING

We first describe our EFS design framework, including the state transition and the mechanism in each state. Then, we describe how we dynamically update the *Division_Factor*, which is a factor used to reduce the backoff interval.

3.1 The EFS Design Framework

We propose an efficient fair scheduling scheme by integrating the ideas of calculating backoff interval of packets and decreasing backoff timer exponentially [10, 16]. The essential idea of the proposed EFS is to choose a backoff interval that is proportional to the finish tag of a packet to be transmitted and to decrease the backoff timer exponentially when consecutive idle slots are detected. We assume that all packets at a node belong to a single flow as proposed in [10]. In a multiple flows case, when station i needs to select the next packet that it will attempt to transmit, it selects the packet with the smallest finish tag among packets at the front of all backlogged flows at station i. When packet P_i^k reaches the front of its queue, it is tagged with a start tag and a finish tag. P_i^k represents the k_{th} packet arriving at the flow at station i. S_i^k, the start tag of P_i^k, is calculated as $S_i^k = v(a_i^k)$, where a_i^k represents the real time when packet P_i^k reaches the front of the flow. Finish tag F_i^k is assigned as follows:

$$F_i^k = S_i^k + Scaling\_Factor \times \frac{L_i^k}{\phi_i} \tag{3}$$

where L_i^k represents the length of packet P_i^k and ϕ_i represents the weight of station i.

An appropriate choice of the *Scaling_Factor* allows us to choose a suitable scale for the virtual time. The next step is to choose a backoff interval such that a packet with smaller finish tag will be assigned a smaller backoff interval. This step will be performed in the *successful packet transmission* state. An active station can be in two modes at each contention cycle, namely, the *transmitting mode* when it wins a contention and the *deferring mode* when it loses a contention. When a station transmits a packet, the result is either successful or failed. Therefore, a station will be in one of the following four states at each contention cycle: a *successful packet transmission* state, a *backoff timer decrement* state, a *transmission failure* state, and a *deferring* state. Fig. 2 shows the state transition of the EFS. The operation of the proposed EFS is described as follows according to the state of a station that it belongs.

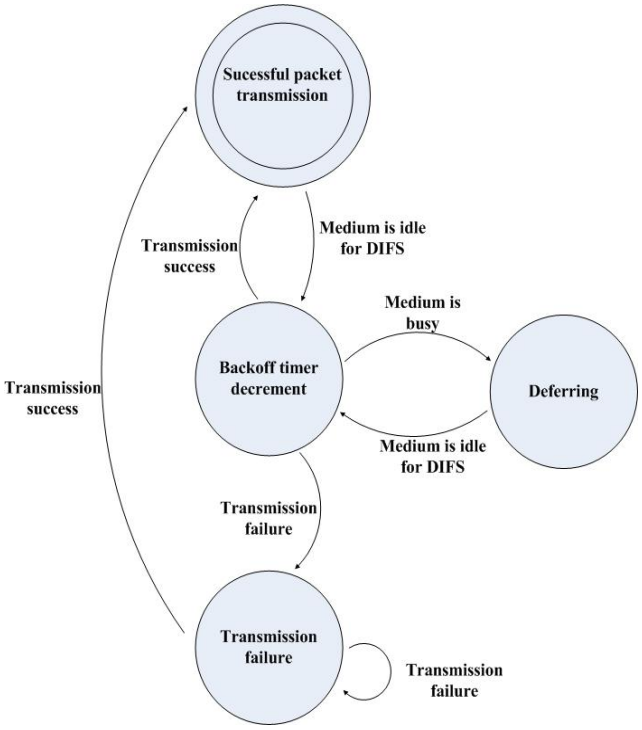

Fig. 2. The state transition of EFS.

3.1.1 *Backoff timer decrement* state

If an active station senses the medium idle for a slot, then it will start to decrease its backoff timer by *aSlotTime*, as shown in Eq. (4):

$$B_{new} = B_{old} - aSlotTime \tag{4}$$

where B_{old} means the old backoff interval and B_{new} means the new backoff interval.

Note that if $B_{new} < aSlotTime$, then $B_{new} = 0$. In addition, if there are *BTD* (*Backoff Threshold*) consecutive idle slots being detected, its backoff timer will be decreased much faster according to Eq. (5):

$$\text{if } DF \neq 1,\ B_{new} = B_{old}/DF \tag{5}$$

where *DF* represents *Division_Factor*. If the value of *DF* equals to 1, its backoff timer will be decreased according to Eq. (4).

Note that *BTD* is a constant parameter which will be clear later and *DF* will be modified dynamically according to the network load condition. The algorithm of modifying *DF* will be explained later. A station can maintain a counter (called *BTD Counter*) whose default value is equal to *BTD*. When the station detects an idle slot time, it decreases its *BTD* counter by one. When the *BTD* counter reaches zero, the station will start

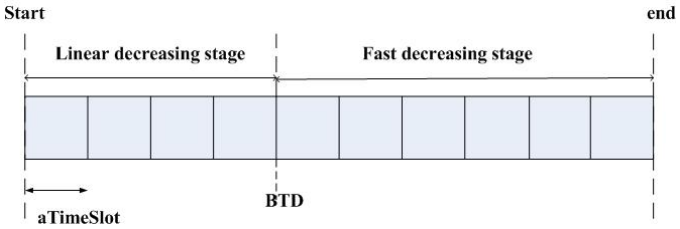

Fig. 3. The backoff timer decrement state.

to decrease the backoff timer exponentially. That is, the station enters the fast decreasing stage from the linear decreasing stage, as shown in Fig. 3.

For example, consider two flows, flow 1 at station 1 with weight 0.1 and flow 2 at station 2 with weight 0.05. Let the packet size be 1000 bytes, *DF* be 1.5, *BTD* be 60, and the *Scaling_Factor* be 0.02. The value of *Scaling_Factor* × *PacketSize/weight* is the backoff interval. The detail of choosing a backoff interval will be illustrated in section 3.1.3. For simplicity, we assume that ρ is 1. Thus, the backoff interval will be 200 time slots for flow 1, and 400 time slots for flow 2. When an idle slot is detected, the backoff interval will be 199 time slots for flow 1, and 399 time slots for flow 2, respectively. Each flow will subtract one slot time from its backoff interval until there are 60 consecutive idle time slots being detected. Suppose that there are 60 consecutive idle slots being detected. Now the backoff interval is 140 time slots for flow 1 and 340 time slots for flow 2. Then when an idle slot is detected, the backoff interval will be 93 (140/1.5) time slots for flow 1 and 226 (340/1.5) time slots for flow 2. When an idle slot is detected again, the backoff interval will be 62 (93/1.5) time slots for flow 1 and 150 (226/1.5) time slots for flow 2. The rest may be deduced by analogy. The backoff interval of flow 1 will reach zero first, and then flow 1 will transmit its packet immediately.

3.1.2 *Transmission failure* (packet collision) state

Each station must maintain a *CollisionCounter* that counts the number of successive collisions. If a station notifies that its packet transmission has failed possibly due to packet collision, then the station must react with the following procedure:

(1) It increases *CollisionCounter* by 1;
(2) It chooses a new Backoff interval uniformly distributed in

$$\left[1,\ \left\lfloor (1+\frac{1}{DF})^{CollisionCounter-1} \times K \right\rfloor\right]$$

where *K* is a constant parameter.

The station will choose a small backoff interval in the range [1, *K*] after the first collision for a packet. The station will choose a backoff interval in the range [1, $\lfloor(1 + 1/DF) \times K\rfloor$] if collision occurs again. To protect against the situation when too many stations collide, the range for the backoff interval grows exponentially with the number of con-

secutive collisions. The station with higher *DF* than others will access the medium with a higher probability. The motivation for choosing a small backoff interval after the first collision is that since the colliding station was a potential winner of the contention for channel access, it is the colliding station turn to transmit in the near future. Therefore, the backoff interval is chosen to be small to increase the probability that the colliding station wins the contention soon.

3.1.3 *Successful packet transmission* (choosing a backoff interval) state

If a station i successfully transmits a packet with finish tag Z at time t, it will pick a suitable backoff interval B_i for its next packet as proposed in [10] and it sets its virtual clock v_i to $\max(v_i(t), Z)$. Suppose the station i will transmit its next packet, P_i^k. The station i will tag the packet with a finish tag. This step is performed at time a_i^k. Thus, station i picks a backoff interval B_i for packet P_i^k, as a function of F_i^k and the current virtual time $v_i(a_i^k)$, as follows:

$$B_i = \left\lfloor F_i^k - v(a_i^k) \right\rfloor. \tag{6}$$

We combine Eq. (3) with Eq. (6) and observe that, since $S_i^k = v(a_i^k)$, Eq. (6) reduces to:

$$B_i = \left\lfloor (Scaling\_Factor \times \frac{L_i^k}{\phi_i}) \right\rfloor. \tag{7}$$

Finally, we will randomize B_i, as shown in Eq. (8). The meanings of L_i^k and ϕ_i have been described before. ρ is a random variable uniformly distributed in the range [0.9, 1.1] to reduce the possibility of collisions. After deciding the value of B_i, we assign B_{old} the value of B_i to preserve B_i which will be used in the *deferring state*. B_{old} is used in the *backoff timer decrement state*.

$$B_i = \left\lfloor \rho \times (Scaling\_Factor \times \frac{L_i^k}{\phi_i}) \right\rfloor \tag{8}$$

In this state, the station must reset its *CollisionCounter* to zero.

3.1.4 *Deferring* state

Each transmitted packet is tagged with a finish tag. So when at time t, station i hears a packet with finish tag Z, it calculates the difference between Z and the current virtual clock $v_i(t)$, *i.e.*, $\Delta = (Z - v_i(t))$. If station i is in the fast decreasing stage with $\Delta > 0$, it resets its B_i according to Eq. (9):

$$\max\{B_{old}, (B_i - \Delta)\}. \tag{9}$$

Then, it resets B_{old} to B_i. The reason we preserve the value of B_i is obvious here. In order to maintain fairness, this procedure is necessary since we incorporate a fast backoff mechanism. Finally, the station i sets its virtual clock v_i equal to $\max(v_i(t), Z)$.

One notable point is why we select *max* in Eq. (9). The consideration is that when a station i receiving a packet just enters the fast decreasing stage, the B_{old} of the station may be greater than $B_i - \Delta$. In this situation, resetting the backoff interval, B_i, to B_{old} is more appropriate.

3.2 Dynamically Updating Division_Factor

In order to take into account the network load condition for *Division_Factor* (DF) adaptation, we used a measurement scheme similar to [17] to get the related network information. We used the number of collisions as an indicator of the network load condition. The time domain is divided into continuous *measurement periods* (*MPs*) with specified period size. An *MP* is defined as the number of time slots. When the kth measurement period, denoted by MP_k, expires, the station summarizes the network load condition indicator $\delta(k)$ [17] during MP_k as follows:

$$\delta(k) = \frac{n_c}{n_s} \tag{10}$$

where n_c is the number of collisions which occurred during MP_k, and n_s is the total number of packets sent during MP_k. Because $\delta(k)$ cannot precisely represent the long-term network load condition, we also used an estimator of *Exponentially Weighted Moving Average* [17] to smoothen the estimated value of each measurement period. The average network load condition indicator during MP_k, denoted by $\delta_{avg}(k)$, is computed as follows:

$$\delta_{avg}(k) = \theta \times \delta_{avg}(k-1) + (1-\theta) \times \delta(k) \tag{11}$$

where $0 < \theta < 1$. We set θ to be 0.8, as proposed in AEDCF [17]. *DF* will be modified in every *MP* according to the following condition:

$$\Delta_k = \delta_{avg}(k) - \delta_{avg}(k-1) \tag{12}$$

$$\text{If } \Delta_k > 0,\ DF = \max(1, (1 - \delta_{avg}(k)) \times DF) \tag{13}$$

$$\text{If } \Delta_k < 0,\ DF = \min(2, (1 + \delta_{avg}(k)) \times DF). \tag{14}$$

So when the network load condition becomes better than that in the last *MP*, we increase the *DF* to decrease the backoff timer faster and when the network load condition becomes worse than that in the last *MP*, we reduce *DF* to decrease the backoff timer slower. Thus the value of *DF* is adapted to the network condition. We let the minimum value of *DF* be 1 so that the backoff timer decrements just like the original procedure (*i.e.*, Eq. (4)). We let the maximum value of *DF* be 2 so that the backoff timer will not decrement too fast so as to increase the collision probability.

4. SIMULATION MODEL AND SIMULATION RESULTS

We evaluated the performance of the proposed EFS using the network simulator, *ns*-2 [18], which supports IEEE 802.11 DCF functionality. We extended the simulator to

implement the proposed EFS. We also included the implementations of DFS [10] and EDCA [13] made by other researchers.

4.1 Simulation Model

In the simulation model, the bandwidth of the wireless LAN is 11 Mbps and the number of stations in the wireless LAN is n. In a wireless LAN with n stations, we set up $n/4$ high priority flows and $n/4$ low priority flows (n is always chosen to be a multiple of 4). Flow i is set up from station i to station $i + 1$ (the stations are numbered 0 through $n - 1$). In the simulations of all three schemes, the high priority traffic flows generate packets with a constant bit rate of 1 Mbps and the low priority traffic flows generate packets with a constant bit rate of 500 Kbps. All flows generate packets with length of 1,000 bytes. Table 1 shows the parameters used in the simulations for comparison of different schemes.

Table 1. Simulation parameters.

Scheme	Parameter	Value
Common parameters	Number of stations	n
	Timeslot length	20μs
	Bandwidth	11 Mbps
	CW_{min}	31
	CW_{max}	1023
	DIFS	50μs
DFS or EFS	$Weight_{high}$	8/3n
	$Weight_{low}$	4/3n
	$CollisionWindow$	4
	$Scaling\_Factor$	0.02
EDCA	$AIFS_{High}$	30μs
	$TxopLimit_{high}$	0.003
	$AIFS_{low}$	50μs
	$TxopLimit_{low}$	0
EFS	BTD	60
	DF	1.3
	K	8
	Measurement period	5,000 time slots

When choosing the parameter settings to use for different schemes, we tried to use settings specified in the standards or papers where the schemes were specified [10, 13, 17]. In the simulation of EDCA, we chose queue 2 and queue 3 with default values [13] so that queue 2 can get bandwidth that is close to 2 times the bandwidth of queue 3. We tested this with 4 stations where 2 flows generated packets with a constant rate of 11 Mbps. Therefore, we assigned queue 2 with weight 3/8n and queue 3 with weight 4/8n. The sum of weights of all flows adds to 1. In the simulation of EFS, we set BTD to 60 and DF to 1.3.

4.2 Simulation Results

We evaluated the *aggregate throughput* of high priority flows by summarizing all the throughput of high priority flows and the aggregate throughput of low priority flows by summarizing all the throughput of low priority flows. We also evaluated the *fairness index* [19]. We also compared the proposed EFS with DFS and EDCA.

4.2.1 Aggregate throughput

Fig. 4 shows the aggregate throughput for low and high priority stations versus the number of stations. We can see that the EDCA's aggregate throughput of high priority flows is higher than EFS and DFS. EDCA can achieve higher throughput because in the EDCA *ns*-2 simulation [13], it has implemented the *Contention Free Burst* (CFB). The EDCA can achieve much higher aggregate throughput with CFB than without CFB. We can also see that the throughput changes between 16 and 24 stations. The throughput degrades because in the case of 16 stations, there are 8 flows whose total offered load are 6 Mbps while in the case of 24 stations, there are 12 flows whose total offered load is 9 Mbps, which are greater than the real bandwidth limit of 11 Mbps in wireless LANs. Thus, in the case of 24 stations, some time was wasted due to contention such as collision or choosing larger backoff intervals. EFS and DFS both can allocate bandwidth to flows in proportion to their weights. The proposed EFS can achieve higher aggregate throughput than DFS because the EFS effectively reduces idle slots using *Division_Factor* (*DF*) and can dynamically change the *DF* value according to the network load. Note that the high priority flows of EDCA got more aggregate throughput than the low ones as the number of stations increased. The EDCA resulted in unfairness between high and low priority flows.

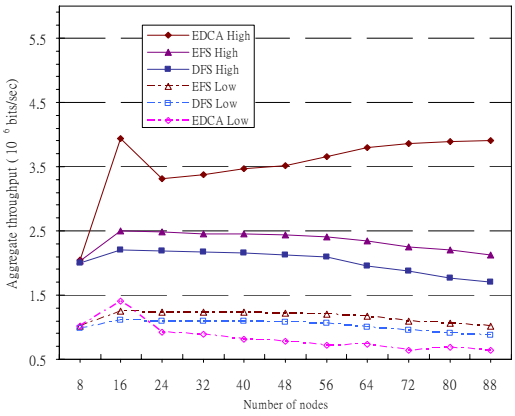

Fig. 4. The aggregate throughput of high and low priority flows.

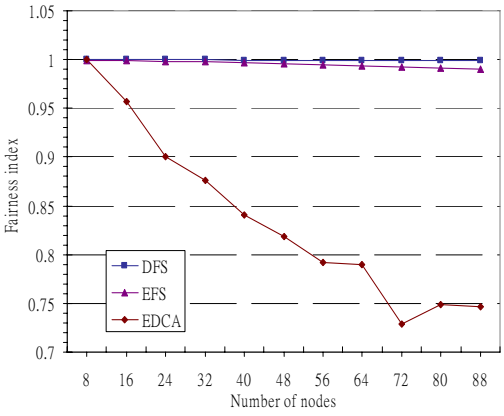

Fig. 5. The fairness index of different schemes.

4.2.2 Fairness index and throughput variance

For environments where all flows are always backlogged, the *fairness index* [19] is defined as follows:

$$fairness\_index = \frac{(\sum_f T_f/\phi_f)^2}{number\_of\_flows \times \sum_f (T_f/\phi_f)^2} \tag{15}$$

where T_f denotes the throughput of flow f, and ϕ_f denotes the weight of flow f. Remind that the higher the value of the fairness index is, the better the fairness is. Fig. 5 shows that the fairness index achieved by EDCA degrades as the number of stations in the wireless LAN increases. This is because that as the number of stations increases, there is an increase of collisions. To be fair, colliding stations should get prior access over other stations after suffering a collision. However, in EDCA, the colliding nodes start binary exponential backoff to pick a larger backoff interval and hence do not get prior access over other stations. This results in unfairness towards the colliding nodes. On the other hand, the proposed EFS and the DFS can achieve a high fairness index even when the number of stations increases.

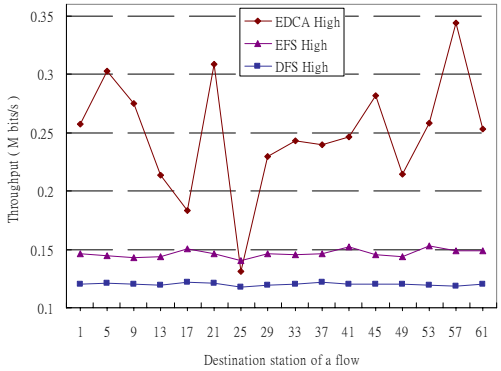

Fig. 6. The throughput obtained by each high priority station (flow).

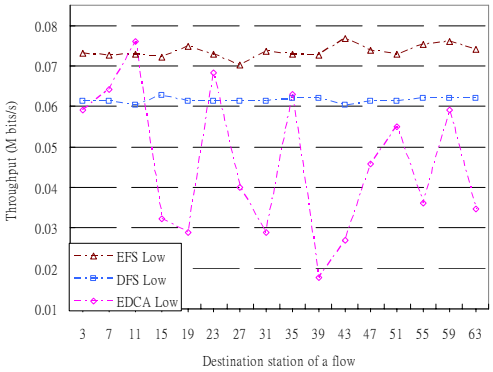

Fig. 7. The throughput obtained by each low priority station (flow).

Fig. 6 and 7 show the throughput achieved by each high priority flow and each low priority flow. The X-axis represents the receiving station ID of each flow. We considered a scenario when there are sixty-four stations. We can see that flows with the same priority in the EDCA suffered from high throughput variance. The reason leading to this situation is just as the reason leading to unfairness, and randomness in choosing a backoff interval is also another significant factor. However, in DFS and EFS, flows with the same priority had low throughput variance. High throughput variance is undesired for time-sensitive applications. We have solved this problem by assigned flows' backoff intervals in proportion to their weights. Note that EFS has a little higher throughput variance than DFS due to its fast backoff mechanism. Table 2 summarizes the qualitative evaluation

Table 2. Qualitative comparison of various schemes.

Parameter	DCF [15]	EDCA [13, 14]	DFS [10]	EFS (proposed)
Fairness	Medium	Low	High	High
complexity	Low	Low	Medium	Medium
MAC delay	High	Medium	High	High (6% less than DFS)
Aggregate throughput	Medium	High (CFB enabled)	Medium	Medium (13% better than DFS)

results by comparing various schemes. In the proposed EFS, high and low priority flows got more aggregate throughput than DFS by 13%. The EFS and DFS can achieve high fairness. High fairness means low throughput variance. Low throughput variance is more suitable for QoS-sensitive applications.

4.2.3 Mac delay

We also evaluated the average MAC delays of the EFS, DFS, and EDCA. The average MAC delay was computed by summarizing the MAC delay of all the packets and averaging it. The results are shown in Fig. 8. We can see that EFS slightly reduced the average MAC delay compared to DFS since we have enhanced the EFS by reducing idle time slots and have also enhanced the collision handling of the EFS. As shown in Eq. (7), the value of DF can determine the rate of reducing idle time slots. The EFS and DFS both had higher average MAC delay than the EDCA due to the use of contention free bursting (CFB) in EDCA. Since the EFS and DFS chose a backoff interval for each packet with consideration of fair scheduling, this resulted in their longer average MAC delays than that of the EDCA.

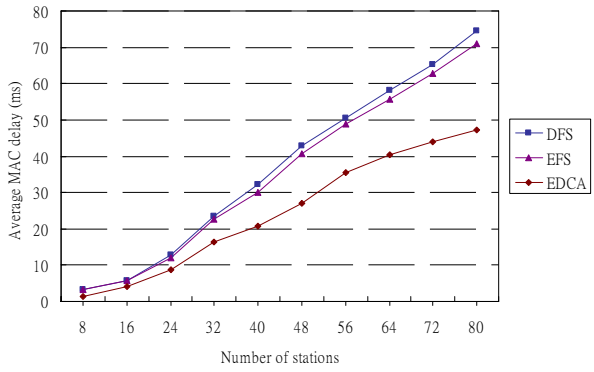

Fig. 8. The average MAC delay of different schemes.

5. CONCLUSIONS

We have proposed an efficient distributed fair scheduling scheme, EFS, for supporting weighted fair scheduling in IEEE 802.11e wireless LANs. The goal of the EFS is

to achieve fair allocation of bandwidth and to enhance the throughput performance. In the EFS, a packet with the smallest ratio between its length and weight receives the highest priority to transmit. The main idea of this scheme is to pick a backoff interval in proportion to the finish tag of the packet. By picking a right backoff interval and using appropriate collision resolution in the transmission failure state, we can achieve fair allocation of bandwidth. With the fast backoff mechanism in the backoff timer decrement state and by dynamically adjusting the backoff interval according to the network load, we can enhance the performance of the EFS. Simulation results have shown that the proposed EFS can achieve 13% higher throughput, 6% lower MAC delay than the DFS. Both approaches have higher fairness indexes than EDCA, which mean that they can allocate bandwidth to flows in proportion to their weights.

REFERENCES

1. W. Pattara-Atikom, P. Krishnamurthy, and S. Banerjee, "Distributed mechanisms for quality of service in wireless LANs," *IEEE Wireless Communications*, Vol. 10, 2003, pp. 26-34.
2. A. Lindgren, A. Almquist, and O. Schelen, "Evaluation of quality of service schemes for IEEE 802.11 wireless LANs," in *Proceedings of IEEE Local Computer Networks*, 2001, pp. 348-351.
3. D. He and C. Q. Shen, "Simulation study of IEEE 802.11e EDCF," in *Proceedings of IEEE Vehicular Technology Conference*, Vol. 1, 2003, pp. 685-689.
4. G. W. Wong and R. W. Donaldson, "Improving the QoS performance of EDCF in IEEE 802.11e wireless LANs," in *Proceedings of IEEE Communications, Computers and Signal Processing Conference*, Vol. 1, 2003, pp. 392-396.
5. M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round-robin," *IEEE/ACM Transactions on Networking*, Vol. 4, 1996, pp. 375-385.
6. S. J. Golestani, "A self-clocked fair queueing scheme for broadband applications," in *Proceedings of IEEE INFOCOM*, 1994, pp. 636-646.
7. P. Goyal, H. M. Vin, and H. Cheng, "Start-time fair queuing: a scheduling algorithm for integrated services switched networks," *IEEE/ACM Transactions on Networking*, Vol. 5, 1997, pp. 690-704.
8. L. Zhang and T. T. Lee, "Performance analysis of wireless fair queuing algorithms with compensation mechanism," in *Proceedings of IEEE International Conference on Communications*, Vol. 7, 2004, pp. 4202-4206.
9. J. Song, Y. Chen, and L. Li, "Simple fair scheduling algorithm for wireless networks," in *Proceedings of IEEE International Conference on Communications*, *Circuits and Systems*, Vol. 1, 2004, pp. 374-378.
10. N. H. Vaidya, P. Bahl, and S. Gupta, "Distributed fair scheduling in a wireless LAN," in *Proceedings of Mobile Computing and Networking Conference*, 2000, pp. 167-178.
11. 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, 1993, pp. 344-357.
12. J. C. R. Bennett and H. Zhang, "Wf2q: worst-case fair weighted fair queueing," in

Proceedings of IEEE INFOCOM, 1996, pp. 120-128.
13. IEEE 802.11 WG, IEEE standard for information technology – Telecommunications and information exchange between systems – LAN/MAN specific requirements - Part 11: Wireless medium access control (MAC) and physical layer (PHY) specifications: Medium access control (MAC) enhancements for quality of service (QoS), IEEE Std. 802.11e, 2005.
14. An IEEE 802.11e EDCF and CFB simulation model for NS-2.26, http://www.tkn.tu-berlin.de/research/802.11e_ns2/.
15. IEEE standard for wireless LAN medium access control (MAC) and physical layer (PHY) specifications, ISO/IEC 8802-11: 1999 (E), Aug. 1999.
16. Y. Kwon, Y. Fang, and H. Latchman, "Fast collision resolution (FCR) MAC algorithm for wireless local area networks," in *Proceedings of Global Telecommunications Conference*, Vol. 3, 2002, pp. 2250-2254.
17. L. Romdhani, Q. Ni, and T. Turletti, "Adaptive EDCF: enhanced service differentiation for IEEE 802.11 wireless ad hoc networks," in *Proceedings of IEEE Wireless Communication and Networking*, Vol. 2, 2003, pp. 1373-1378.
18. The Network Simulator – ns-2, http://www.isi.edu/nsnam/ns, 2002.
19. R. Jain, A. Durresi, and G. Babic, "Throughput fairness index: an explanation," ATM Forum Document No. ATM_Forum/99-0045, Feb. 1999.

Kun-Nan Tseng (曾昆南) received the B.S. degree in the Department of Applied Mathematics and the M.S. degree in the Department of Computer and Information Science from National Chiao Tung University, Taiwan, in 2002 and 2004, respectively. He is currently an engineer in the Information & Communications Research Labs of Industrial Technology Research Institute. His research interests include quality of service, mobile computing, wireless Internet, and wireless protocols.

Kuochen Wang (王國禎) received the B.S. degree in Control Engineering from National Chiao Tung University, Taiwan, in 1978, and the M.S. and Ph.D. degrees in Electrical Engineering from the University of Arizona in 1986 and 1991, respectively. He is currently a Professor in the Department of Computer Science, National Chiao Tung University. From 1980 to 1984, he was a Senior Engineer at the Directorate General of Telecommunications in Taiwan. He served in the army as a second lieutenant communication platoon leader from 1978 to 1980. His research interests include wireless (ad hoc/sensor) networks, mobile computing, and power management for multimedia portable devices.

Hung-Cheng Shih (施宏政) received the B.S. degree and the M.S. degree in the Department of Computer and Information Science from National Chiao Tung University, Taiwan, in 1997 and 2002, respectively. He is currently a Ph.D. student in the Department of Computer Science, National Chiao Tung University. His research interests include mobile computing, wireless Internet, 3G telecommunication systems, and power management.