STUDY OF DISTRIBUTED FAIR SCHEDULING IN WIRELESS LOCAL AREA NETWORKS

A Thesis

by

SEEMA GUPTA

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2000

Major Subject: Computer Science

STUDY OF DISTRIBUTED FAIR SCHEDULING IN WIRELESS LOCAL AREA NETWORKS

A Thesis

by

SEEMA GUPTA

Submitted to Texas A&M University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Approved as to style and content by:

Nitin H. Vaidya (Chair of Committee)	Riccardo Bettati (Member)
A. L. Narasimha Reddy (Member)	Wei Zhao (Head of Department)

August 2000

Major Subject: Computer Science

ABSTRACT

Study of Distributed Fair Scheduling in Wireless Local Area Networks. (August 2000)

Seema Gupta, M.Sc., Indian Institute of Technology

Chair of Advisory Committee: Dr. Nitin H. Vaidya

A Fair Scheduling policy is required to support differentiated QoS requirements of contending flows in a wireless channel. This thesis presents a study of the Distributed Fair Scheduling (DFS) algorithm proposed for wireless Local Area Networks. The thesis evaluates DFS protocol and studies the unfairness in IEEE 802.11 standard.

The wireless channel capacity varies with time and location due to the presence of location-dependent wireless errors. Therefore, the error-free scheduling specification is not sufficient for fair allocation of channel capacity amongst contending flows in an error-prone wireless channel. This thesis borrows the idea of dynamic weight adjustment from prior work and applies it in the Distributed Fair Scheduling algorithm to provide long-term fairness in the presence of wireless errors.

To my mother, father, brother and sisters

ACKNOWLEDGMENTS

I would like to express my sincere thanks to my advisor Dr. Vaidya for his guidance, comments and encouragement. Without his advice, enthusiasm and help, my graduate school experience would not have been so fulfilling.

I would like to thank Dr. Victor Bahl of Microsoft Research for his guidance and comments during the course of work.

I thank Dr. Bettati and Dr. Reddy for being part of my advisory committee.

I would like to thank Ms. Bhaskaran for her help during the course of work.

I would like to thank NSF and Sun Microsystems for supporting me financially to carry out my research.

I would like to thank my parents for their constant encouragement and support.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Fairness in wireless mediums is a challenging problem. A fair scheduling policy is required to support differentiated Quality of Service (QoS) requirements of different applications. Some applications such as voice, are delay-sensitive and require a guaranteed share of bandwidth allocated to them. Video applications can tolerate losses and can adapt to variations in available bandwidth. To account for such differentiated QoS requirements, fair scheduling policies introduce a notion of weights associated with each flow. The weight of a flow is a measure of the dynamic share of bandwidth the scheduler promises to allocate to the flow. The goal of a fair scheduler is to allocate bandwidth in proportion to the weights of the flows.

Most of the proposed fair scheduling policies [1], [8], [9], [18], assume a centralized scheduler that has information about the backlog of all flows. This thesis relates to the study of fairness in a wireless Local Area Network (LAN). The IEEE 802.11 [10] wireless LAN standard specifies a Distributed Co-ordination Function that follows a distributed implementation. 802.11 does not have any provision for weights and does not account for variable length packets. 802.11 is unfair over short time scales, and is biased towards flows with large packet sizes. A Distributed Fair Scheduling (DFS) protocol for a wireless LAN is presented in [20]. DFS allocates bandwidth to flows in proportion to their weights and accounts for variable packet sizes.

DFS schedules packets for transmission based on their eligibility. Because of the distributed nature of the protocol, there may be collisions, which cause priority reversal and affect the fairness achieved. This thesis presents a study of the DFS

[0]The journal model is *IEEE Transactions on Automatic Control.*

protocol and fairness in DFS.

It is difficult to quantify fairness in the wireless medium due to the presence of wireless errors. Wireless errors vary with time and location. Therefore, the error-free scheduling specification is not sufficient for fair allocation of channel capacity amongst contending flows in an error-prone wireless channel. This thesis borrows the idea of dynamic weight adjustment from [5], and applies it to the Distributed Fair Scheduling algorithm. A flow lagging due to wireless errors can reclaim lost bandwidth by dynamically adjusting weights. Administrative controls can be exercised to limit the amount of compensation allowed to erroneous flows by means of a power factor proposed in [5]. This thesis borrows the idea of distinguishing losses due to errors and losses due to collisions from [21]. Simulation results show that long-term fairness can be achieved by erroneous flows in DFS with dynamic adjustment of weights.

The reminder of the thesis is organized as follows. Chapter II describes the basic DFS protocol presented in [20]. It presents the motivation for this thesis, and describes the related work. Chapter III describes the details of the simulation scenarios and parameters used. It briefly describes the performance of DFS presented in [21] for the sake of completeness. Chapter IV presents a study of fairness in the DFS protocol and suggests an enhancement to 802.11 to improve its fairness. Chapter V evaluates the performance of DFS with dynamic weight adjustment in the presence of wireless errors. Chapter VI gives the conclusions and scope for future work.

CHAPTER II

BACKGROUND

IEEE 802.11 [10] is a Medium Access Control (MAC) protocol that uses Carrier Sense Multiple Access with Collision Avoidance. IEEE 802.11 standard specifies that the sender would transmit a short Request-To-Send (RTS) frame and wait for a Clear-To-Send (CTS) frame from the receiver before transmitting the data frame. These frames contain information about the length of the data packet to follow. This enables the neighbors overhearing the packet to backoff for that duration and prevent a collision with the on-going transmission of the data packet. The receiver sends a short Acknowledgement (ACK) frame upon the successful receipt of the data packet.

A. IEEE 802.11 Distributed Co-ordination Function

In the Distributed Co-ordination Function of the IEEE 802.11 standard [10], [20], a node i wishing to transmit a packet chooses a backoff interval of B_i slots. The backoff interval is a random variable that is uniformly distributed over an interval $[0, cw]$, where cw is the size of the contention window. When the channel becomes idle for a *difs* [10] period, node i starts decrementing B_i by one after each slot time. If the channel becomes busy, it freezes the backoff timer B_i and restarts it when the channel becomes idle for a *difs* period again. When B_i becomes zero, node i sends an RTS to the intended destination. If two nodes choose the same backoff interval and start counting down together, their RTSs will collide. Then the colliding nodes choose new backoff intervals and repeat the process of contention. After the successful receipt of an RTS, the receiver sends back a CTS to the sender. The sender transmits data after getting the CTS and the receiver sends back an ACK after receiving the data packet reliably.

B. Distributed Fair Scheduling Protocol

This section summarizes the basic DFS protocol presented in [20] to lay the background for further understanding of the protocol in later chapters.

DFS chooses backoff intervals based on the length of the packet and the weight of the flow. When a node i with weight w_i wishes to transmit its k-th packet of length l_i^k, it chooses a backoff interval B_i as,

$$B_i = \left\lfloor Scaling\_Factor * \frac{l_i^k}{w_i} \right\rfloor \tag{2.1}$$

Scaling_Factor allows for the choice of a suitable scale for the backoff intervals. To reduce the possibility of collisions, the authors propose a randomization of B_i as follows,

$$B_i = \lfloor \rho * B_i \rfloor \tag{2.2}$$

where ρ is a random variable with mean 1. B_i thus obtained is referred to as the initial backoff interval.

DFS separates the backoff intervals used initially from those used after collision. When a collision occurs for node i, it chooses a new backoff interval as follows. *CollisionWindow* and *MaxCollision* are constant parameters.

- *CollisionCounter* is incremented by 1.
- if $CollisionCounter < MaxCollision$ then, a variable x is chosen uniformly distributed in $\left[1, 2^{CollisionCounter-1} * CollisionWindow\right]$, and B_i is chosen to be the smallest prime larger than x. Otherwise, $cw = 2 * cw + 1$ and B_i is chosen to be a random number uniformly distributed in the interval $[0, cw]$.

The above procedure of collision resolution in DFS chooses a relatively small B_i after collision. The rationale behind the choice of a small window after collision as explained by the authors, is that the initial backoff intervals represent the eligibility of the packet. Since a packet met with a collision, it should be given preference by assigning a small B_i after collision. However, B_i grows exponentially with the number of consecutive collisions to protect against the situation when many nodes collide.

The DFS protocol allocates throughput in proportion to the weights of the flows. It accounts for variable packet sizes and variable weights. Chapter III describes the performance of DFS in comparison to 802.11 as presented in [21].

C. Motivation

DFS can be implemented with simple modifications to IEEE 802.11. A motivation behind this work is to understand short-term fairness in DFS. DFS provides fair allocation of channel bandwidth in the error-free environment. It fails to provide fair allocation in the presence of wireless errors. The wireless medium is typically characterized by location-dependent errors [17]. Any fair scheduling scheme for wireless environments that does not account for errors may not be useful in error-prone wireless environments. Hence, this work also derives motivation to provide fairness in DFS in the presence of wireless errors.

D. Related Work

A lot of fair scheduling policies have been presented for a centralized scheduler in a wired network. The ideal Generalised Processing Scheduler (GPS) [12], [18], schedules bits at a time based on the weights of the participating flows in a round-robin order. Let $S_i(t_1, t_2)$ be the amount of service flow i receives in the interval $[t_1, t_2]$. Suppose

the weights of flows i and j are w_i and w_j respectively. If flow i is backlogged during the interval $[t_1, t_2]$ then, the following condition holds:

$$\frac{S_i(t_1, t_2)}{S_j(t_1, t_2)} \geq \frac{w_i}{w_j} \ , \qquad \forall j \tag{2.3}$$

Equality holds in the above equation when flow j is also backlogged during the interval $[t_1, t_2]$.

Packetized approximation of the GPS scheduler is proposed in [1], [8], [9], based on the notion of start time, virtual time and finish time.

Since the wireless channel is characterised by location-dependent errors, the algorithms of fair allocation for a wired link cannot to applied directly to the wireless link. A description of the issues and approaches of fair queueing in wireless networks is presented in [4]. Many fair queueing algorithms for fair scheduling in a wireless link have been discussed in [3], [13], [14], [16]. All of these adopt the centralised approach and follow a compensation procedure. They maintain state information about the lead and lag of flows. When a flow is unable to utilize the channel due to wireless errors, another error-free flow is allocated the channel. This causes the former to lag and the latter to gain a lead. The compensation model in all these schemes enables the lagging flows to make up for the loss and, allows for the graceful degradation of leading flows. A generic approach to provide compensation to lagging flows is discussed in [19], which maintains an additional compensating flow at each node to allocate additional bandwidth to the lagging flows. This approach can be easily applied in the distributed wireless network environment as discussed in [21].

Effort-Limited Fair (ELF) scheduling for wireless networks is presented in [5]. ELF proposes a novel notion of effort-limited fairness for wireless links by extending the centralised weighted fair queuing algorithms [1], via dynamic weight adjustment. ELF guarantees that all flows experiencing an error-rate below a per-flow threshold

receive their expected service. This thesis borrows the idea of dynamic weight adjustment from ELF [5] to provide long-term fairness in DFS in the presence of wireless errors.

CHAPTER III

SIMULATION DETAILS

A. Simulation Scenario and Parameters

This section describes the simulation scenario and DFS parameters used in the simulations. The scenario and parameters used in this work are identical to those used in [21]. This work uses a DFS protocol implementation on *ns-2* simulator [7], [15], based on the implementation in [21][1]. The channel bandwidth is considered to be 2Mbps. The simulation environment consists of n number of nodes. All nodes are stationary and are in the transmission range of each other to simulate a broadcast LAN. The maximum number of nodes considered is 128. The number of nodes is always even. On a LAN with n nodes, $n/2$ flows are set up. A flow i is established from node i to node $i+1$. In this thesis identical flows refer to flows which are always backlogged and have equal weights and equal packet sizes. Unless specified otherwise, the following assumptions are made:

- All flows are always backlogged.
- All flows use CBR traffic.
- The duration of the simulation is 6 seconds.
- All data packets are 512 bytes. Since 802.11 does not account for variable packet sizes, constant length packets are considered in DFS for its comparison with 802.11.

[1]There are small differences in implementation of DFS in this work and in [21], specifically in the implementation of backoff interval.

- *Scaling_Factor* is 0.02. The choice of the *Scaling_Factor* governs the trade-off between aggregate throughput and fairness as described in [21].
- *CollisionWindow* is 4 slots. A larger value is needed for collision resolution in the presence of wireless errors as described in Chapter V.
- There are n nodes with $n/2$ identical flows of weight $2/n$. Since 802.11 does not account for weights, equal weights are considered in DFS for its comparison with 802.11.
- The MAC header is counted towards the throughput calculation, to account for the MAC overhead associated with the transmission of each data packet. Therefore, the MAC frame size of 584 bytes is considered for the throughput calculation in DFS and 802.11.
- The random variable ρ in Equation 2.2 is uniformly distributed in the interval $[0.9, 1.1]$.
- *MaxCollision* is set to 3.
- The sum of weights of flows adds to 1.

The variable L_i^k in Equation 2.1 refers to the MAC frame size which defaults to 584 bytes. Plugging the default values in Equation 2.2, the backoff interval B_i is given by the following for n identical flows.

$$B_i = \left\lfloor \frac{0.02 * 584}{\frac{2}{n}} \right\rfloor, or B_i = \lfloor 5.84 * n \rfloor \tag{3.1}$$

Since the variable ρ in Equation 2.2 is uniformly distributed in the interval $[0.9, 1.1]$, the randomization interval of B_i for n identical flows varies within 20%

of B_i in Equation 3.1 and, the interval can be reduced to the following for the default values,

$$\lfloor 5.256 * n \rfloor <= B_i <= \lfloor 6.424 * n \rfloor \tag{3.2}$$

Note that the initial backoff interval B_i is directly proportional to the number of flows for the case of identical flows. It varies inversely with the weight of a flow.

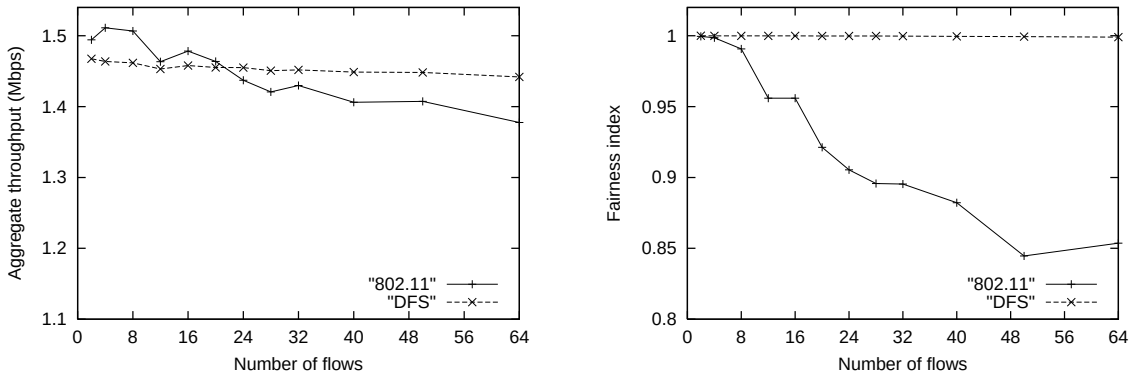
The following throughput fairness index presented in [11] is used for the environments where all flows are always backlogged. Suppose T_f denotes the throughput achieved by flow f and w_f denotes the weight of flow f, then the throughput fairness index is calculated as,

$$\text{fairness index} = \frac{\left(\sum_f T_f/w_f\right)^2}{\text{number of flows} * \sum_f (T_f/w_f)^2} \tag{3.3}$$

Larger the fairness index value, larger is the fairness measure. A value of 1 represents 100% throughput fairness. This is the fairness metric used throughout this thesis.

B. Collision Resolution in 802.11 and DFS

The default initial contention window size specified in 802.11 [10] is 31. This value is also used for the 802.11 simulation results presented in this thesis. This is also the minimum contention window value specified in 802.11. 802.11 uses binary exponential backoff for collision resolution. After every collision, 802.11 doubles the contention window size (cw) and resets it to the minimum window size only after a successful transmission. 802.11 doubles the window size as, $cw_{i+1} = 2 * cw_i + 1$. Therefore, with consecutive collisions the window grows from 31 to 63, 127, 255, 511 and 1023. 1023



(a) Aggregate throughput (b) Fairness index

Fig. 1. Comparison of DFS with 802.11

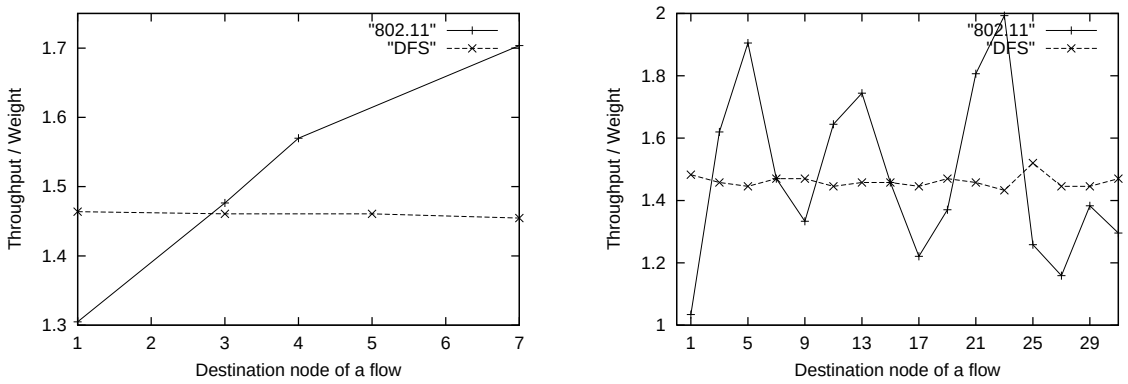
is the maximum limit on the contention window size.

DFS reacts to collisions unaggressively initially by picking a small window. With *CollisionWindow* equal to 4 slots, DFS chooses a contention window of 5 slots after the first collision. Upon consecutive collisions the contention window grows from 5 to 11, and 17 initially, by adjusting it to the nearest prime number for a uniform distribution of backoff intervals. With further subsequent collisions, the contention window grows exponentially from 35, to 71, 143, 287 and so on, as explained in Chapter II. This value is limited by 1023 slots. As shown in Chapter IV, the contention window after collision in DFS usually does not grow to larger values.

C. Performance of DFS

This section repeats the performance comparison of DFS with 802.11 as described in [21] for the sake of completeness and for the understanding of following chapters in this thesis.

Figure 1a plots the variation in aggregate throughput as the number of flows in-



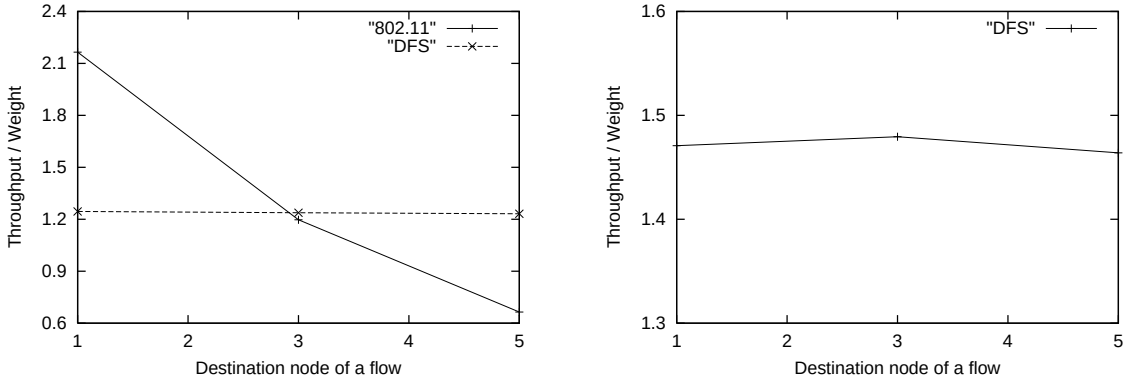
(a) Four flows (b) Sixteen flows

Fig. 2. Distribution of bandwidth amongst flows

creases. The remarkable characteristic of DFS is that, the aggregate throughput does not degrade as the number of flows increases. Such a constant aggregate throughput curve is highly desirable of any scheme that should scale well with the number of flows.

Figure 1b plots the variation of fairness index as the number of flows increases. In 802.11, the fairness index decreases as the number flows increases. This is because of increased collisions due to increased load on the channel. On the other hand DFS gives 99.99% throughput fairness even for large the number of flows. This is also a desirable property of any fair scheduling policy that is expected to scale with the number of flows. This is a measure of long-term fairness as it is over a duration of 6 seconds and is averaged over 4 runs.

Figure 2 plots the distribution of bandwidth amongst contending flows. Since a fair allocation of bandwidth should allocate throughput in proportion to the weights of the flows, the y-axis plots $Throughput/Weight$. A fair allocation would result in a horizontal straight line curve for the $Throughput/Weight$ metric. In Figure 2a there are 4 identical flows and DFS allocates equal bandwidth to all the flows as desired.



(a) Variable packet sizes (b) Variable weights

Fig. 3. Long-term fairness with variable packet sizes and variable weights

Figure 2b considers the case of 16 identical flows. DFS allocates equal bandwidth to all the flows, whereas 802.11 exhibits unfairness by allocating unequal bandwidth amongst flows.

Figure 3a shows fairness achieved by DFS when variable packet sizes are used. In Figure 3a, there are three flows with constant packet sizes of 584, 328 and 200 bytes respectively. DFS allocates equal share of bandwidth to all the three flows. 802.11 does not account for variable packet sizes and is biased towards flows with larger packet sizes.

In Figure 3b shows fairness achieved by DFS when variable weights are assigned to the flows. It considers three flows which have equal packet sizes of 584 bytes. The weights of the three flows are 0.9, 0.05 ad 0.05 respectively. DFS allocates bandwidth in proportion to the weights of the flows as observed by the straight line curve for the $Throughput/Weight$ metric. 802.11 does not have any provision for variable weights. Hence, a corresponding curve for 802.11 is not plotted.

The key observations in this section can be summarized as follows,

- Aggregate throughput obtained in DFS scales with the number of flows. It does not degrade with increase in load as observed in 802.11.
- DFS achieves almost 100% long-term fairness even with large number of flows.
- DFS allocates bandwidth in proportion to the weights of the flows and accounts for variable packet sizes.

CHAPTER IV

FAIRNESS IN DFS

This chapter presents a comparison between the contention behavior of 802.11 and DFS. It discusses the unfairness in 802.11. The use of a small window after collision is justified, and the significance of choosing weights in DFS is discussed. This chapter presents a comparison of DFS with GPS [18]. This chapter discusses the fast convergence of fairness index in DFS and studies short-term fairness in DFS.

A. Fairness in 802.11

As discussed in [2], 802.11 can lead to unfair allocation of bandwidth. Due to binary exponential backoff, 802.11 reacts aggressively to collisions by doubling the contention window with each collision and resetting it to the minimum contention window value only upon a successful transmission. This causes large variations in the backoff counter. This can lead to unfairness when flows with relatively large backoff counters collide with the relatively less backed off flows, which in turn win contention, maintaining a small backoff counter as discussed in [2]. To solve this problem [2], proposes a mild backoff mechanism to share the contention information correctly and have the backoff counters correctly reflect the level of contention on the channel. DFS solves this problem by separating the initial backoff counter from the backoff counter used for collision resolution. It chooses the initial backoff counter value proportional to the number of flows.

The following scenario studies the impact of collisions in 802.11 and DFS. When multiple identical flows start together, they all wait for an interval of *difs* [10] to sense the channel as idle, before they transmit the *RTSs*. These *RTSs* collide and all flows backoff in both 802.11 and DFS.

In DFS with 16 identical flows which always have something to send, all of the flows experience collisions initially and choose a small backoff interval in the range [1,5] with the parameters specified in Chapter III. Since 16 flows randomly pick backoff intervals in such a small range, more than two flows choose the same interval with a high probability, leading to collisions. They go into exponential backoff, choosing backoff intervals from a larger range of [1,11]. This time packets from fewer flows collide. They again choose backoff intervals from a larger range of [1,17]. Eventually, flows stagger apart. When such a large number of identical flows start together, there are collisions initially but the synchrony is destroyed due to randomization and, the flows stagger apart. Subsequent and later collisions do no usually involve more than two flows.

In comparing the same scenario with 802.11, all the nodes wait for *difs* interval to sense the channel as idle and transmit their *RTSs* in the same slot, leading to collisions. This happens only at the start of the simulation. All flows go into exponential backoff doubling their window sizes. After collision all the flows choose backoff intervals in the range of [0,63]. Subsequently, there are fewer collisions and the colliding flows pick a backoff interval in a larger range of [0,127]. For the same scenario in DFS, collisions are resolved at a lower contention window value of 17 only. 802.11 reacts to collisions very aggressively by shooting up its window size to up to 1023, which is the maximum limit on the window size. As the number of flows increases, the contention window values in 802.11 get very large.

A comparison of contention window values after collision in 802.11 and in DFS is given in Figure 4 for a simulation with 50 identical flows. It is observed that the number of collisions in 802.11 is relatively more than in DFS. This is because of the choice of large initial backoff intervals in DFS as opposed to a small contention window of 31 in 802.11. The curve for DFS plots the backoff intervals used after colli-

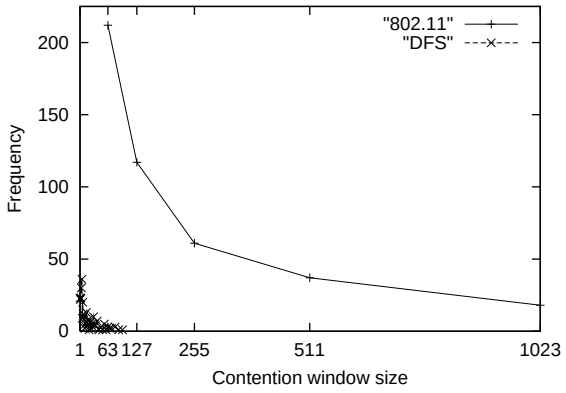


Fig. 4. Distribution of contention window values after collision in 802.11 and DFS

sion, whereas the points on the 802.11 curve represent the contention window values. Therefore, the envelop of the two curves should be used for comparison. Figure 4 shows that 802.11 reacts to collisions very aggressively and increases the contention window exponentially to 1023. DFS begins with large enough initial backoff intervals in the range of [525-642] obtained from Equation 3.2. Such a large backoff interval reduces collisions. At the same time, the choice of a small window after collisions increases the number of collisions, but not as much as in 802.11. In this case, DFS gets slightly higher throughput than 802.11 and much higher long term throughput fairness index than 802.11.

The aggressive reaction of 802.11 to collisions has serious impact on fairness. In two different simulations with 16 and 50 identical flows, it was observed that some flows were completely backed off during certain intervals. Clearly, aggressive binary exponential backoff causes unfairness in 802.11.

B. Contention Behavior of 802.11 and DFS

It is important to analyze the difference in the contention behavior of 802.11 [10] and DFS [20]. 802.11 uses *fixed* size initial contention windows. With each collision in

802.11, the contention window is doubled, and it is reset to the minimum contention window value only after a successful transmission. On the other hand DFS chooses initial backoff interval of *variable* size, which is proportional to the number of flows. Upon collision, DFS chooses a small contention window in a fixed interval. With subsequent collisions, the interval size is increased exponentially.

When the number of flows is large, the contention behavior of 802.11 and DFS are opposite of each other. Suppose there are 50 flows. 802.11 will choose initial contention window sizes of 31. Since the number of flows is large, the contention window values after collisions will shoot to up to 1023 as observed in Section A. Whereas in DFS, the initial backoff intervals are chosen in a larger range of [525-642] as described in Section A. After collision it uses small contention window values which grow from 5 to up to 143 as explained in Chapter III. In other words, 802.11 chooses small windows initially and large windows after collision. On the other hand, DFS chooses large backoff intervals initially and small intervals after collisions.

Since the initial backoff intervals in DFS are large enough, being proportional to the load on the channel, the choice of a small window after collision does not necessarily deteriorate its performance. In 802.11, choosing a small backoff interval after collision would aggravate contention on the channel and degrade the effective utilization of the channel. DFS chooses initial backoff intervals inversely proportional to the weights of the flows. This not only reduces the probability of collisions but also distributes bandwidth in proportion to the weights. Since DFS intelligently picks appropriate (depending on the number of flows) contention window values initially, it can react less aggressively to collisions. Choosing small contention window after collision allows a lagging flow to make up for the loss. In other words, it prevents the 802.11 phenomenon discussed in Section A, by preventing already backed-off flows from being put into further backoff.

C. Analysis of Collision Resolution in DFS

This section presents an analysis of unaggressive collision resolution in DFS. It justifies the use of a small window after collision by comparing different variations of DFS with 802.11 and emphasizes the significance of the choice of weights in DFS.

802.11 does not have any notion of weights and does not account for variable packet sizes. Therefore, for an effective comparison between 802.11 and DFS, fixed weights were assigned to the flows in DFS so as to be comparable with the initial window size of 31 in 802.11. Two variants of DFS are considered. The first one is DFS1, wherein weights of flows is fixed to 0.33. The rationale behind choosing a fix weight of 0.33 is that, this leads to an initial window size of approximately 31 slots. In DFS1 the actual backoff intervals are chosen in the range of [31-38], obtained from Equation 3.2. This gives a small range of 8 slots only as compared to the range of 31 slots chosen in 802.11. Hence, a second and more refined variant of DFS considered here is DFS2. In DFS2, the weights of flows is fixed to 0.066. This leads to initial backoff intervals in the range of [159-194]. The length of this randomization interval is equal to 36 slots, which is comparable to the randomization interval of 31 slots used in 802.11. DFS1 and DFS2 follow the collision resolution mechanism of DFS.

Figure 5 shows the increase in the sum of weights of all flows with the number of flows in DFS1, DFS2 and in DFS. In DFS weights are always adjusted to get a sum of 1.

Figure 6a compares the aggregate throughput with the number of flows in DFS1 and DFS2. Figure 6b compares the fairness index with the number of flows in DFS1 and DFS2.

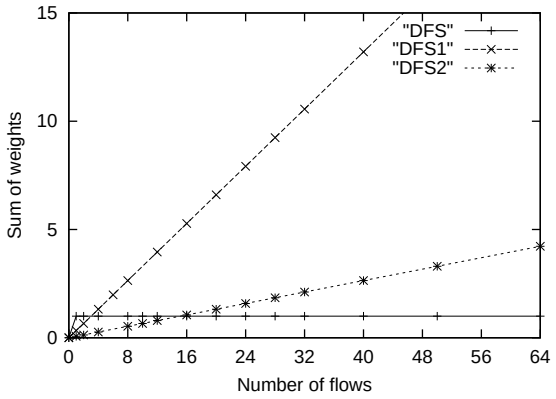
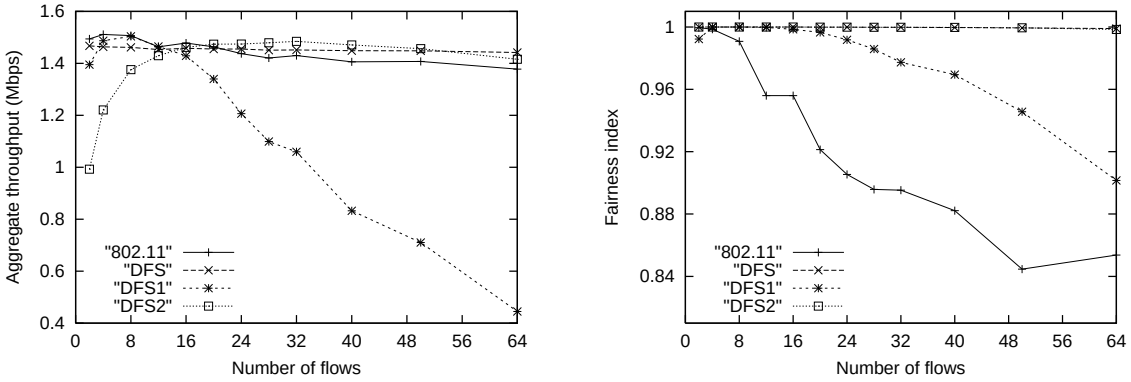


Fig. 5. Sum of weights versus number of flows



(a) Aggregate throughput (b) Fairness index

Fig. 6. Degradation in throughput and fairness index due to small contention windows

In DFS1, when the number of flows is 2, a weight of 0.33 yields larger windows than a weight of 0.5 in DFS, calculated as $1/number\ of\ flows$. Therefore, aggregate throughput of DFS1 is lower than the aggregate throughput in DFS for 2 flows. For 4 and 8 flows, DFS1 chooses smaller backoff intervals of the order of 31, whereas DFS chooses larger intervals, as they are proportional to the number of flows. Therefore, DFS1 achieves higher aggregate throughput without much degradation of fairness index because the load due to 4 and 8 flows is relatively less. When the number of flows increases beyond 12, there is a significant drop in aggregate throughput and in long-term fairness index achieved by DFS1. Sum of the weights of more than 12 flows is greater than 4 as shown in Figure 5. For more than 12 flows, since the initial backoff interval is smaller, and backoff after collision is not aggressive enough, it leads to degradation in throughput and fairness in DFS1 as observed in Figure 6.

In DFS2, when the number of flows is less than 16, DFS2 attains very low throughput in comparison with DFS and 802.11, because of the choice of unnecessarily large initial backoff intervals. Since the weights of DFS and DFS2 are same for 16 flows, their throughput and fairness curves cross over at 16. When the number of flows increases from 20 to 40, DFS2 gets advantage of choosing smaller windows in comparison to DFS. It attains higher throughput and fairness index. When the number of flows increases beyond 50, throughput and fairness of DFS2 degrades, because the initial backoff interval in the range of [159-194] is not large enough to sustain the load of such a large number of flows and, the choice of a small window after collision leads to increased contention. The weights of 50 flows adds up to 3.3 as shown in Figure 5. Clearly, if the number of flows is increased further, the performance of DFS2 will degrade further. As observed, the aggregate throughput and fairness index in DFS2 are lower for 64 flows.

It is evident that the choice of small window after collision deteriorates perfor-

mance when the weights of the backlogged flows add up to a larger number. This threshold can be set at 3. DFS achieves a nearly constant aggregate throughput and 99.99% long-term fairness, because the sum of the weights add up to 1. This enables the use of appropriate initial window sizes and, the choice of a small window after collision does not deteriorate performance. If the weights add up to a number larger than one, then choosing a small window after collision would deteriorate performance as studied in this section.

D. Comparison of DFS with GPS

In order to compare DFS with GPS, a non-work-conserving GPS scheduler is described in [21]. This scheduler is referred to as NW-GPS. The NW-GPS scheduler transmits data according to GPS scheduling. It idles whenever the wireless channel is transmitting control information like, RTS, CTS or ACK frames. It also idles during the transmission of the MAC header and during collisions. Whenever the DFS scheduler successfully transmits data, the NW-GPS scheduler serves data from amongst the backlogged flows, and it idles at all other times. As discussed in [21], this will give a comparison of fairness only and it does not account for the throughput comparison.

A simulation study was done for 2, 4 and 16 flows. The simulation duration was 10 seconds. All flows were always backlogged and had equal weights. All flows had equal sized packets of 512 bytes for simplicity.

The service time of a packet is the time at which the scheduler finishes servicing the packet. Since DFS is a packetized service discipline and all packets are constant size, the difference in service time between DFS and NW-GPS is $n * T$, where T is the transmission time for a packet and n can be 0,1,2,3....

Figure 7 compares DFS and NW-GPS scheduling disciplines for two flows. The

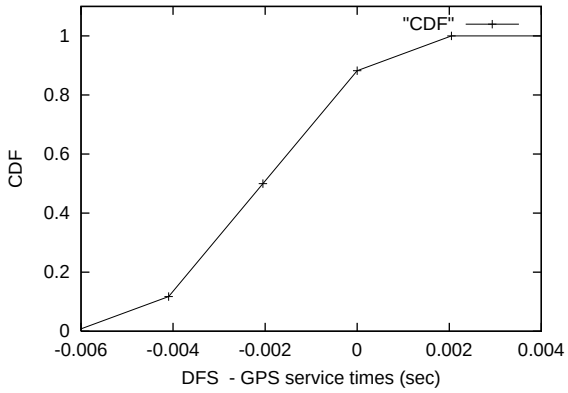
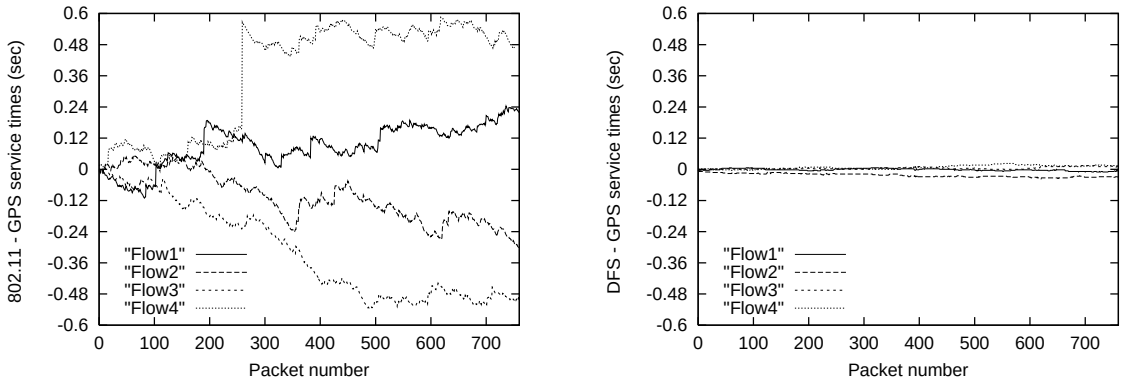


Fig. 7. CDF function for difference in service time of DFS and NW-GPS for two flows

x-axis represents the difference in service time of DFS and NW-GPS. The Y-axis represents the cumulative probability distribution $p(t <= T)$, that is the probability that the difference in service time of DFS and NW-GPS schedulers is less than equal to T time units. A negative value on the x-axis implies that DFS is ahead of NW-GPS by x time units and a positive value implies that DFS lags behind NW-GPS by x time units. With the parameters specified in Chapter III, the transmission time of one packet is 0.002048 seconds. Figure 7 shows that DFS is ahead of NW-GPS by at most two packets and it lags behind NW-GPS by at most one packet service time. Therefore, in this case the difference in the service time of DFS and NW-GPS is bounded by the transmission time of two packets in the error free environment.

Figure 8 compares the difference in service time of 802.11 and DFS with NW-GPS for four flows. This difference clearly relates to the short-term fairness. Since 802.11 is unfair over the long term as studied in Section A, it cannot be fair in the short term. This is also observed in Figure 8a.

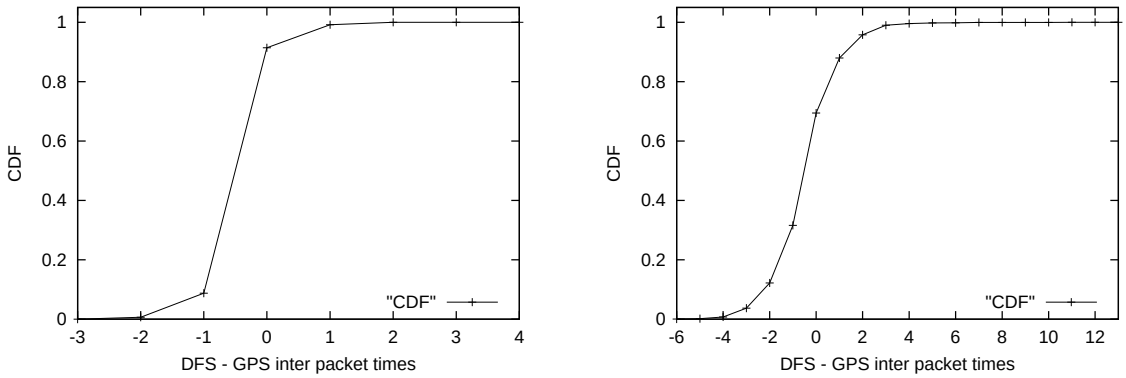
Due to collisions, DFS does not always service packets in order of their eligibility. This causes reordering of packets and leads to a skew in the difference in service time of DFS and NW-GPS. The skew caused by the out of order packets adds up



(a) 802.11 (b) DFS

Fig. 8. Comparison of difference in service time of 802.11 and DFS with NW-GPS for four flows

cumulatively with subsequent out of order packets. This phenomenon gets more prominent as the number of flows increases, because collisions, and number of out of order packets increase with the number of flows. This phenomenon is observed in Figure 8b, which plots the difference in service time of DFS and NW-GPS for four flows. This graph is scaled for comparison with 802.11. This skew in service times can be illustrated through an example. Let the difference in service time of DFS and NW-GPS be denoted as ΔT. Suppose DFS schedules out of order packets due to collisions during an interval [t1,t2]. This will lead to differences in the service time of packets in DFS and in NW-GPS. Later on during [t3,t4], DFS schedules packets in the same service order as that in NW-GPS. In this case ΔT during [t3,t4] will not be 0, as desired, instead it will be equal to the ΔT during [t1,t2]. Therefore, it is useful to compare the difference in *inter packet* service time of DFS and NW-GPS to eliminate the cumulative effect of previous out of order packets from being carried further. This yields a comparison of relative service times instead of the absolute service times. Since MAC overhead is ignored in the comparison, the service order is



(a) Four flows (b) Sixteen flows

Fig. 9. Difference in inter packet service time of DFS and NW-GPS

more important than the absolute service time.

Figure 9 plots the difference in inter packet service time between DFS and NW-GPS for all flows. Since 802.11 is unfair over the long-term, a similar plot for the comparison of short-term fairness in 802.11 is not given. The x-axis represents the difference in inter packet service time between DFS and NW-GPS in terms of number of packets. For instance a value of -2 represents that DFS serviced packets earlier by two packets transmission time (the packet sizes are constant for these simulations). Analogously, a positive value of 2 on the x-axis represents that DFS was lagging behind GPS by two packets transmission time. The y-axis represents the cumulative probability $p(t <= T)$, that the difference in inter packet service time of DFS and NW-GPS schedulers is less than equal to T time units.

Figure 9a plots the CDF function for 4 flows. As observed, the probability that DFS gets ahead of NW-GPS or lags behind NW-GPS by more than 2 packets is 0. Figure 9b plots the CDF function for 16 flows. For this simulation, DFS does not get ahead of NW-GPS by more than 6 packets and it does not lag behind NW-GPS by more than 12 packets.

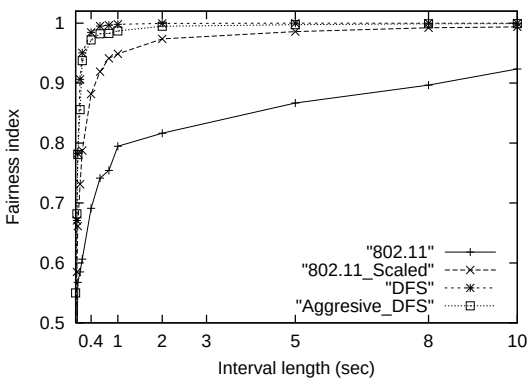
Since DFS algorithm is non-deterministic due to priority inversion caused by collisions [21], it is difficult to derive deterministic bounds for the difference in service time of DFS and NW-GPS. However, the simulation results in this section show that DFS order does not differ from the GPS order by large values.

E. Short-term Fairness in DFS

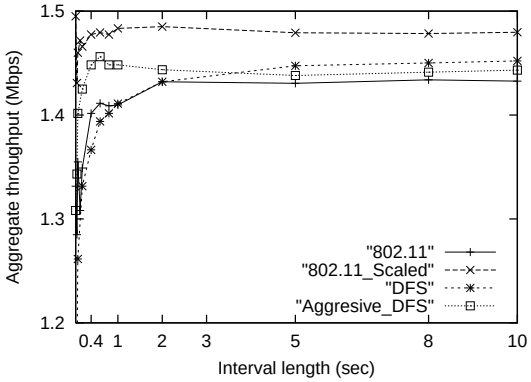
DFS uses variable sized backoff intervals proportional to the number of flows for fair allocation of bandwidth. This idea can be applied to 802.11. A variation of 802.11 is considered where the initial window sizes are proportional to the number of flows as in DFS. Clearly, such a variant of 802.11 can be compared with DFS only when the weights and packet sizes of the contending flows are equal. This variant of 802.11 is referred to as 802.11_Scaled.

To compare the effect of unaggressive collisions in DFS, another variant of DFS is considered which performs aggressive collision resolution using binary exponential backoff. This is referred to as Aggressive_DFS. Collision resolution is said to be aggressive in Aggressive_DFS because, after collision it doubles its contention window instead of choosing a small window.

Figure 10 gives a comparison between 802.11, 802.11_Scaled, DFS and Aggressive_DFS for a simulation with twenty four identical flows. Fairness index is calculated using the Equation 3.3. Aggregate throughput is calculated by adding the throughput of all the twenty four flows. This data is for a single run. The x-axis represents the interval over which fairness index and throughput are calculated. It shows the rate of convergence of fairness index and throughput for the four schemes. Figure 11 magnifies the [0-1] second interval in the Figure 10 for a better comparison of short-term convergence of fairness index and aggregate throughput.

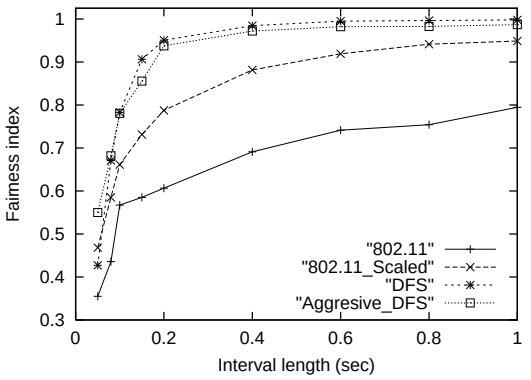


(a) Fairness index

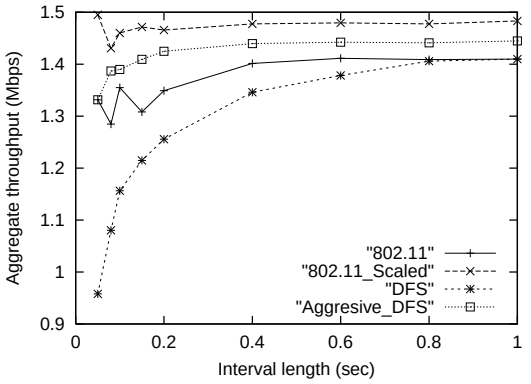


(b) Aggregate throughput

Fig. 10. Long-term convergence of fairness index and aggregate throughput



(a) Fairness index



(b) Aggregate throughput

Fig. 11. Short-term convergence of fairness index and aggregate throughput

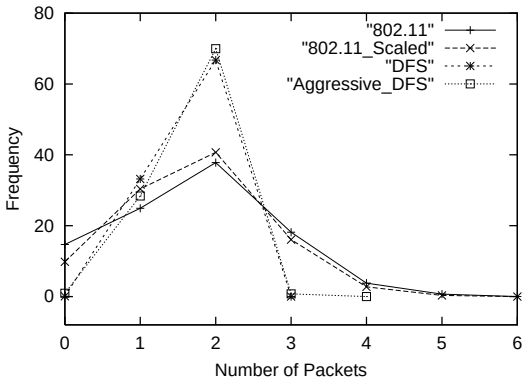
It is observed that 802.11_Scaled attains higher fairness index and aggregate throughput in comparison to 802.11. It converges to the long-term fairness index faster than 802.11. The improvement in its fairness is due to the choice of proportionally large initial window sizes based on the number of flows. It performs poorer than DFS and Aggressive_DFS because, it chooses the initial contention window in a larger range of [0-308] as opposed to a shorter range of [252-308], obtained from Equation 3.2, used by DFS and Aggressive_DFS. This causes large variations in the backoff intervals. Moreover, the choice of exponentially large windows after collision does not allow for the compensation of lagging flows and hence, fairness is lower.

Aggressive_DFS achieves lower aggregate throughput in comparison to DFS because of the choice of large backoff intervals after collision. It also attains lower fairness index as compared to DFS which shows that the choice of a small window after collision in DFS, improves its fairness by giving the lagging flows preference in channel allocation.

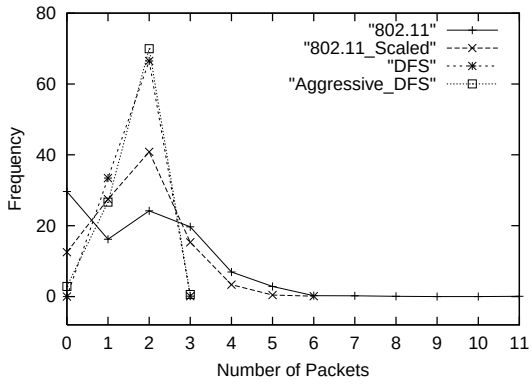
Clearly, DFS performs best amongst all the schemes with fastest convergence of fairness index. Performance of 802.11_Scaled suggests that significant improvement in the performance of 802.11 can be achieved by using initial window sizes proportional to the number of flows.

To study the short-term fairness in DFS, the number of packets received in very short sliding window intervals were considered. The length of the interval was scaled with the number of flows. Figure 12a plots the frequency distribution of the number of packets received in a sliding window of 20msec for 4 flows in 802.11, 802.11_Scaled, DFS and Aggressive_DFS. The window slides by half the length of the interval in each case. Figure 12b, Figure 13a and Figure 13b give a similar plot for 8, 16 and 24 flows respectively.

It is observed that 802.11 and 802.11_Scaled obtain zero throughput with high

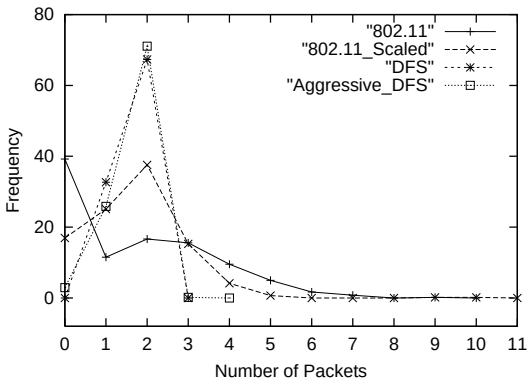


(a) Four flows

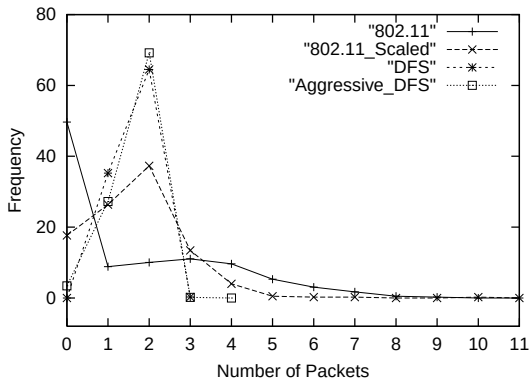


(b) Eight flows

Fig. 12. Frequency distribution of number of packets received in 20msec and 40msec intervals for 4 and 8 flows respectively



(a) Sixteen flows



(b) Twenty four flows

Fig. 13. Frequency distribution of number of packets received in 80msec and 120msec intervals for 16 and 24 flows respectively

frequency. These are the intervals when some flow is completely backed off as discussed in Section A of this chapter. As seen in Chapter III, the fairness index of 802.11 decreases with the number of flows. Similarly, the frequency of zero packets obtained during specified intervals in 802.11 increases with the number of flows. 802.11_Scaled achieves improved short-term fairness in comparison to 802.11 due to a lesser spread in its curve. Larger spread in Figure 12 and Figure 13 indicates poorer short-term fairness. DFS always receives 1 or 2 packets only. This implies that, all flows get either 1, or 2 packets in all intervals considered. DFS receives 0 or 3 packets in all intervals with a frequency less than 0.09%. Aggressive_DFS receives 1 or 2 packets with a higher frequency. It receives 0 or 3 packets with a frequency less than 4%. Aggressive_DFS receives 0 or 3 packets with a higher frequency as compared to DFS. This shows that the choice of a small window after collision in DFS helps in achieving short-term fairness.

CHAPTER V

WIRELESS ERRORS

A. Introduction

Wireless channels are prone to errors due to many factors, such as path loss, multipath interference, attenuation, co-channel interference, background noise and movement of transmitter or receiver. The presence of errors causes wireless link capacity to vary in time and location. This causes difficulty in bandwidth allocation in the presence of wireless errors. The error-prone flows do not get their expected share of bandwidth as per the error-free bandwidth allocation schedule. Additional measures are required to meet the demands of the error-prone flows.

This thesis borrows the idea of dynamic adjustment of weights of flows for fair allocation of bandwidth in the presence of errors from ELF [5]. ELF introduces a novel notion of *effort-outcome* differentiation. When a flow suffers moderately low error rates, it can increase its *effort* by increasing its weight, to make up for the bandwidth lost due to errors. In this case the flow manages to get its expected *outcome*. But when the error rate is high, it should decrease its demand by limiting its weight to prevent deterioration of link utilization. In this case its *effort* is limited and its *outcome* is less than desired.

ELF introduces an interesting concept of power factor. Power factor is a per-flow threshold that can be assigned at the time of admission control. The significance of power factor lies in the fact that, it determines the limit up to which an erroneous flow can try to increase its weight to reclaim the lost bandwidth. A larger value of power factor gives more preference to an erroneous flow at the time of compensation for lost bandwidth. Power factor is useful in two aspects. Firstly, different applica-

tions differ in their capability to tolerate losses. For instance, a video application can tolerate higher degrees of losses than a voice application. Therefore, by assigning a larger value of power factor to a voice application and a lower value to a video application, the amount of compensation allowed to the two applications can be controlled administratively as discussed in [5]. Secondly, compensation for an erroneous flow is at the cost of reduced throughput for other error-free or low-error flows. Power factor of an error-prone flow determines the limit up to which other error-free flows will be affected during the compensation of the error-prone flow.

Following algorithm is used for dynamic adjustment of weights by error-prone flows. Suppose there are n flows sharing a wireless link. Each flow has a weight w_i, power factor p_i and experiences an error rate of e_i. The adjusted weight of the flow i as defined in [5] is,

$$a_i = minimum(\frac{w_i}{1-e_i}, p_i * w_i) \tag{5.1}$$

Suppose the error-free capacity of the wireless link is C. In the error-free condition, flow i with weight w_i gets $C * w_i$ share of bandwidth, which is in proportion to its weight. In the error-prone condition, the channel capacity for flow i drops to $C * (1 - e_i)$. The flow i adjusts its weights to a_i and its share of bandwidth T_i is given by,

$$T_i \;=\; C * (1-e_i) * a_i = \begin{cases} C * w_i, & \text{when } e_i < \frac{p_i - 1}{p_i} \\ C * (1-e_i) * p_i * w_i, & otherwise \end{cases} \tag{5.2}$$

As studied in Chapter IV, the sum of weights of the active flows should not be much larger than 1. This implies that power factor cannot be assigned arbitrarily. It

should be assigned such that the sum of the adjusted weights of all the active error-prone and error-free flows does not exceed a threshold in the worst case of high error rates. This threshold can be set to 2 or 3 based on the observations in Chapter IV.

The error rate experienced by a flow i may vary over time. Let flow i observe an error rate of $obs\_e_i^j$ during interval j. Flow i estimates its error rate e_i^j during interval j by using an averaging function as given in Equation 5.4. The interval j is the interval between errors. Whenever flow i identifies a loss of a packet due to error, it counts the number of packets it received without errors since the last receipt of an errorneous packet. This comprises of interval j. Let the number of data packets received by flow i without errors during interval j be $pkts_i^j$. Then the observed error rate $obs\_e_i^j$, and the estimated error rate e_i^j are given by,

$$obs\_e_i^j = 1/pkts_i^j \tag{5.3}$$

$$e_i^j = \alpha * e_i^{j-1} + (1 - \alpha) * obs\_e_i^j \tag{5.4}$$

The smoothing factor α determines the interval and the rate at which an error-prone flow receives its compensation. A larger value of α, would enable an error-prone flow to reclaim its lost bandwidth farther in the future, allowing for longer periods of compensation. A smaller value of α would be effective for improved fairness only when the estimation of error rate is accurate. Since transient estimated error rates can be very high, limiting the adjusted weight by the power factor, a larger value of α is used in the simulations. This would also improve long-term fairness in the presence of bursty errors by allowing for longer periods of compensation.

B. Error Model

Not much work has been done to analyze the nature of wireless errors. In [6] a typical wireless in-building WaveLan LAN is analyzed to show that WaveLan has very good error characteristics and it can achieve fairly low error rates. In [17], a trace-based simulation of wireless errors is studied in different scenarios. It shows that the error rates are below 10% for typical in-building scenarios. The stationary mobile node scenario studied in [17] is of interest to this work as stationary nodes have been considered throughout this work. The error model for the stationary node scenario can be modeled as a random variable uniformly distributed in a range of low error rates. This error model is used in Section D. The bursty component of error is observed with increased mobility, whereas random low errors rates are observed in the case of in-building stationary mobile nodes.

Most of the work in fair scheduling in wireless transmission in [3], [13], [14], [16], and others consider a centralized scheduler that swaps slots to allow an error-free flow to transmit when an eligible flow experiences error. The losing flow is later compensated. Such swapping of transmission turns and compensation is difficult in the distributed environment of DFS because, flows do not have information about the state (erroneous or error-free, backlogged or unbacklogged) of other flows. When a flow experiences losses due to errors, it will have to explicitly convey this information to the other flows, so that an error-free flow can transmit in place of an erroneous flow and later relinquish the extra bandwidth obtained due to the erroneous flow. Co-ordination of this information is difficult in a distributed environment.

C. Estimation of Error Rate

The goal is to distinguish between losses due to errors and losses due to collisions, so that appropriate backoff measures can be taken. When a flow considers a loss to be due to errors, it can increase its weight dynamically to demand extra bandwidth, but the increase will be limited by a power factor. On the other hand, when it considers the loss to be due to collision, it should exercise a backoff mechanism to lower the contention on the channel.

As discussed in [21], losses can be distinguished based on the following heuristic. The loss of an RTS or a CTS frame is assumed to imply a loss due to collision, and the loss of a DATA or an ACK frame is assumed to imply a loss due to errors. Such a heuristic is conservative and it does not cause increased contention even when the prediction is wrong. For instance, the sender detects the loss of an RTS frame when its CTSTimeout [10] timer expires. This could be either due to loss of RTS, or due to loss of CTS frame. RTS is a small frame and is less probable to be in error as compared to a larger DATA frame. It is the RTS that collides when two or more flows count down their backoff counters simultaneously. This heuristic will fail when an RTS frame is lost due to errors. In that case, collision resolution measures will be taken, thereby reducing the load and therefore, this assumption is conservative. On the other hand, the loss of a DATA frame is unlikely to be due to collision in the absence of mobility. Therefore, this assumption is justified in this scenario.

Since DFS chooses a small contention window after collision, this can lead to increased contention on the channel. This may lead to repeated retransmission requests by the sender as a result of losses due to errors. Therefore, the backoff mechanism in the presence of errors should be aggressive. This is solved by choosing a larger value of the *CollisionWindow* parameter for the simulations. An alternative collision

control scheme discussed below can be employed. The collision resolution mechanism may choose a small contention window after collision as in DFS. It can retain the previous contention window value that resolved collisions and adjust it dynamically. The example of consecutive collisions in Section A of Chapter IV for 16 identical flows is considered. It is observed that the contention window starts with 5, then it increases to 11, and it resolves at 17. In this way it learns the appropriate value (17 here) by gradually incrementing the window after each consecutive collision. A revised backoff mechanism based on [2] can be used. It can learn the appropriate value by retaining the previous contention window value and adjust it dynamically. Then any subsequent collision will start with a contention window of 17, which was learnt from previous experience. If it was successful, it can reduce the value by 1, otherwise increase it by a milder factor of 1.2. A factor of 1.2 is sufficient because it will go into exponential backoff if the collisions become consecutive.

D. Performance

The version of DFS implementing the algorithm discussed in this chapter is referred to as DFS_ERR. This section presents a comparison of DFS_ERR with DFS. DFS_ERR allocates equal share of bandwidth to identical flows subject to similar error conditions. Hence, it is fair after compensation for errors is done. Unless specified otherwise, following default values of parameters is used for the simulation results presented in this section.

- Error rate is a uniformly distributed random variable in a range specified later.
- DFS *CollisionWindow* parameter is equal to 10.
- The maximum number of transmission attempts of an RTS is 3. The maximum number of transmission attempts of a DATA frame is 2.

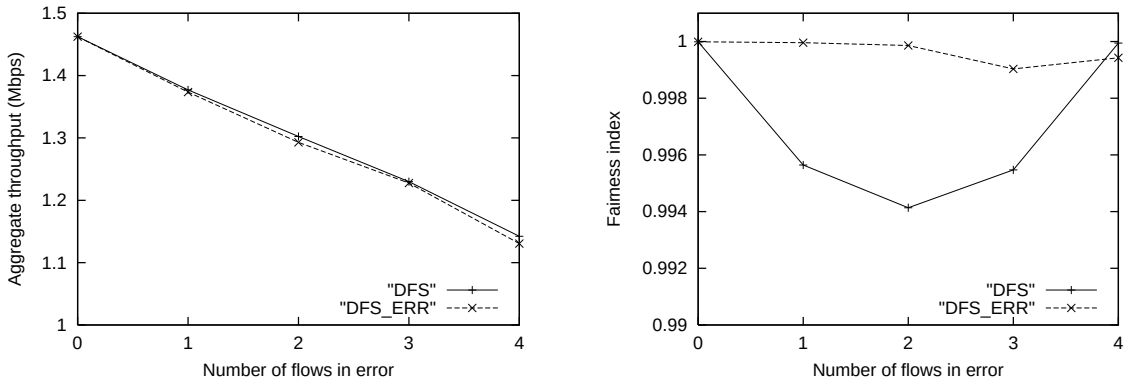
- Power factor is 1.5 and averaging parameter α is 0.8.
- Simulation duration is 4 seconds.

The error model uses packetised approximation of the number of bytes in error. The errors vary uniformly in the interval $[ErrorRate_{low}, ErrorRate_{high}]$. $MaxSize$ is equal to the length of application data measured in bytes. $MaxSize$ is equal to 512 bytes as specified in Section D of Chapter III. The error model is used to mark the error flag of a packet. It chooses a random number $ErrRate$, uniformly distributed in $[ErrorRate_{low}, ErrorRate_{high}]$ interval. The packet is marked to be erroneous with a rate equal to $ErrRate * Size\ of\ Packet/MaxSize$.

Since DFS chooses a small window after collision and after error detection, it implicitly chooses a larger weight for the retransmission of a packet. Therefore, the difference in the performance of DFS and DFS_ERR is not very significant for low error rates.

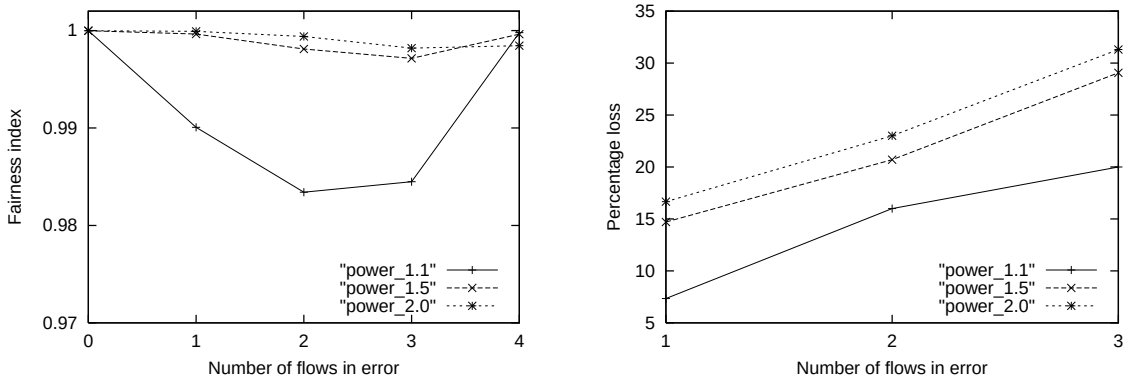
In Figure 14, four flows have been considered. This is a case of location-dependent errors where error-free and error-prone flows co-exist. The x-axis represents the number of flows in error. Figure 14a plots the variation in aggregate throughput with the increase in the number of erroneous flows. Figure 14b plots the variation in fairness index with the increase in the number of erroneous flows. It is observed that both DFS and DFS_ERR are fair when zero or all flows are subject to the same error rate. DFS is most unfair when 50% flows are in error. DFS_ERR achieves improved fairness.

Figure 15a and b show that the amount of compensation allowed to erroneous flows can be controlled by means of power factor. In this case higher error rates varying between [30-40]% have been considered. Figure 15a shows that a lower power factor of 1.1 does not allow for the desired compensation of erroneous flows. Figure 15b



(a) Aggregate throughput (b) Fairness index

Fig. 14. Fairness index and aggregate throughput with 10-25% errors rate for four flows



(a) Fairness index (b) Compensation

Fig. 15. Limitation of compensation by means of power factor

plots the percentage loss suffered by an error-free flow in order to compensate for the erroneous flows. Higher values of power factor lead to higher fairness but at the cost of higher degradation of error-free flows as observed in Figure 15b. This example shows that the amount of bandwidth of the error-free flows utilized for compensation can be controlled by the suitable choice of power factor.

CHAPTER VI

CONCLUSION

This thesis studies the unfairness in 802.11 and presents a study of short-term fairness in DFS by comparing it with a non-work-conserving GPS scheduler described in [21]. It is shown that the difference between the scheduling order of DFS and GPS is small for the simulations studied. It is hard to give strict bounds on the difference in DFS and GPS scheduling orders, because of non-deterministic nature of scheduling and priority inversion caused by collisions in DFS.

Chapter IV highlights an important difference between the contention behavior of DFS and 802.11. It shows that for a large number of flows, their behavior is the opposite of each other. 802.11 chooses a fixed size small initial contention window with aggressive collision resolution whereas, DFS chooses a variable sized large initial backoff interval with unaggressive collision resolution.

Chapter IV justifies the use of a small window after collision in DFS by fixing weights in DFS to be comparable with 802.11 contention window values. It shows that the choice of proportionally large initial backoff intervals compensates for the choice of a small window after collision. This does not deteriorate performance as long as the sum of the weights of active flows does not exceed larger than 1. The choice of a small window after collision is the key to fairness in DFS. DFS achieves almost constant aggregate throughput with 99.99% fairness which scales with the number of flows in the error-free environment. This is a very desirable characteristic of any distributed fair scheduling scheme.

Chapter V describes the scheme for dynamically varying weights to provide long-term fairness in the presence of wireless errors. It distinguishes between losses due to errors and losses due to collisions, to take appropriate measures. Long-term fair-

ness can be achieved in DFS with the presented scheme in the presence of location-dependent variable rate errors. Simulation results show that an erroneous flow can increase its effort to compensate for the bandwidth lost due to errors. The compensation amount and rate can be controlled administratively by the suitable choice of simulation parameters.

Future work can focus on the study of short-term fairness of the proposed scheme in the presence of errors. The issues relating to power conservation in DFS can be studied. Future work can also focus on the application of DFS to provide fairness in mobile adhoc networks.

REFERENCES

[1] J. C. R. Bennett and H. Zhang, "WF2Q: Worst-case fair weighted fair queuing," *In Proc. IEEE INFOCOM Conf.*, vol. 1, pp. 120-128, March 1996.

[2] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, "MACAW: A media access protocol for wireless LANs," *In Proc. ACM SIGCOMM Conf.*, vol. 1, pp. 212-225, August 1994.

[3] V. Bharghavan, S. Lu, and T. Nandagopal, "A unified architecture for the design and evaluation of wireless fair queuing algorithms," *In Proc. ACM/IEEE Mobicom Conf.*, vol. 1, pp. 132-142, August 1999.

[4] V. Bharghavan, S. Lu, and T. Nandagopal, "Fair queuing in wireless networks: Issues and approaches," *IEEE Personal Communications,* vol. 6, pp. 44-53, February 1999.

[5] D. Eckhardt and P. Steenkiste, "Effort-limited fair (ELF) scheduling for wireless networks," *In Proc. IEEE INFOCOM Conf.*, vol. 3, pp. 1097-1106, March 2000.

[6] D. Eckhardt and P. Steenkiste, "Measurement and analysis of the error characteristics of an in-building wireless network," *In Proc. ACM SIGCOMM Conf.*, vol. 1, pp. 243-254, August 1996.

[7] K. Fall and K. Vardhan, "*ns* notes and documentation," Technical Report, VINT Project, University of California, Berkeley and Lawrence Berkeley National Laboratory (LBNL), 1997

[8] S. J. Golestani, "A self-clocked fair queuing scheme for broadband applications," *In Proc. IEEE INFOCOM Conf.*, vol. 2, pp. 636-646, March 1994

[9] P. Goyal, H. M. Vin, and H. Cheng, "Start-time fair queuing: A scheduling algorithm for integrated services packet switching networks," *In Proc. IEEE/ACM Transactions on Networking,* vol. 5, pp. 690-704, October 1997.

[10] IEEE, "IEEE Std 802.11: Wireless LAN medium access control and physical layer specifications," June 1997.

[11] R. Jain, G. Babic, B. Nagendra, and C. Lam, "Fairness, call establishment latency and other performance metrics," Technical Report ATM_Forum/96-1173, ATM Forum Document, Ohio State University, Columbus, August 1996.

[12] S. Keshav, *An Engineering Approach to Computer Networking*, 2nd ed. Reading: Addison Wesley, 1997.

[13] S. Lu, V. Bharghavan, and R. Srikant, "Fair scheduling in wireless packet networks," *In Proc. ACM SIGCOMM Conf.*, vol. 1, pp. 63-74, September 1997.

[14] S. Lu, T. Nandagopal, and V. Bharghavan "A wireless fair service algorithm for packet cellular networks," *In Proc. ACM/IEEE MobiCom Conf.*, vol. 1, pp. 10-20, October 1998.

[15] "Network Simulator", URL:http://www-mash.cs.berkeley.edu/ns/ns.html [Accessed May 2000].

[16] T. S. Ng, I. Stoica, and H. Zhang, "Packet fair queueing: Algorithms for wireless networks with location-dependent errors," *In Proc. IEEE INFOCOM Conf.*, vol. 3, pp. 1103 -1111, March 1998.

[17] B. Noble, M. Satyanarayan, G. Nguyen, and R. Katz, "Trace-based mobile network emulation," *In Proc. ACM SIGCOMM Conf.*, vol. 1, pp. 14-18, September 1997.

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

[19] P. Ramanathan and P. Agrawal, "Adapting packet fair queueing algorithms to wireless networks," in *In Proc. ACM/IEEE MobiCom Conf.*, vol. 1, pp. 1-9, October 1998.

[20] N. H. Vaidya and P. Bahl, "Fair scheduling in broadcast environments," Technical Report, MSR-TR-99-61, Microsoft Research, Redmond, Washington, August 1999.

[21] N. H. Vaidya, P. Bahl, and S. Gupta, "Distributed fair scheduling in a wireless LAN," Technical Report, Department of Computer Science, Texas A&M University, College Station, April 2000.

VITA

Seema Gupta received her Integrated Master of Science degree from Indian Institute of Technology, New Delhi, India in May 1998. She joined the graduate program in Computer Science at Texas A&M University, in September 1998. Her research has been in mobile computing and fairness in wireless local area networks. Her address is Department of Computer Science, Texas A&M University, College Station, TX 77843-3112.

The typist for this thesis was Seema Gupta.