Effort-limited Fair (ELF) Scheduling for Wireless Networks

David A. Eckhardt and Peter Steenkiste
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
davide+@cs.cmu.edu, prs+@cs.cmu.edu

Abstract—
While packet scheduling for wired links is a maturing area, scheduling of wireless links is less mature. A fundamental difference between wired and wireless links is that wireless media can exhibit substantial rates of link errors, resulting in significant and unpredictable loss of link capacity. This capacity loss results in a special challenge for wireless schedulers. For example, a Weighted Fair Queue (WFQ) scheduler assumes an error-free link and specifies how flows should share the link capacity. However, this specification is not sufficient to determine the correct outcome when link capacity is sharply reduced, because flows that have been allocated the same weights may differ greatly in their ability to tolerate throughput loss.

In this paper, we first describe the wireless scheduling challenge in terms of an *effort-outcome disconnection*. **Next we propose a novel notion of fairness for wireless links,** *effort-limited fairness* **(ELF), which extends WFQ via dynamic weight adjustments. ELF guarantees that all flows experiencing an error rate below a per-flow threshold receive their expected service, defined as a specified rate for reserved flows or a specified share of best-effort capacity for best-effort flows. After motivating and defining ELF, we present a practical approximation algorithm, which we evaluate through both trace-driven simulation and measurement of a prototype wireless radio network based on the WaveLAN physical layer.**

Keywords—**Wireless networks, Quality of Service, Scheduling**

I. Introduction

Wireless media can exhibit high, variable error rates that affect network users in a number of ways. One problem is that many applications and end-to-end transport protocols perform very poorly when many packets are lost due to link errors. This problem has been widely studied and proposed solutions include modifying end-to-end protocols [1], [2], techniques such as TCP snooping to avoid undesirable interactions with TCP [3], and the use of local error recovery to hide the link errors from the upper layer protocols [4], [5], [6]. A second problem is that link errors reduce the useful throughput of the link. While a variety of techniques such as swapping time slots between stations [7], [8] or adaptive forward error control [9] can reduce the amount of lost link capacity, some errors will still result in variable link capacity.

The dynamic capacity of wireless links creates a variety of problems for both users and network administrators. The obvious example is supporting reservations. When the capacity of a link drops significantly, it may no longer be possible for the link to meet its commitments for reserved bandwidth, raising the question of which flows should be short-changed. Assume we use a Weighted Fair Queueing (WFQ) scheduler to allocate bandwidth to three flow classes (e.g., audio, video, Web). If the link capacity drops by 50%, a simplistic response would be to proportionally distribute the remaining capacity (and thus the loss). While this may seem fair according to a traditional WFQ model, a network administrator could argue that video is a luxury and should bear the brunt of the capacity loss, i.e., the audio and web flows should get preferential treatment when bandwidth becomes scarce. Similarly, if we consider a wireless network with location-dependent errors, we must decide how much extra air time to assign to high-error stations at the expense of others.

These problems all center on the question of how the scheduler should respond to capacity loss. Since capacity loss is a central issue in wireless link scheduling, it is critical to devise scheduling abstractions and mechanisms that explicitly consider and address this issue. We can best capture the requirements for wireless schedulers by listing properties that the scheduler should meet. The results in this paper are guided by the following list of principles: (1) in an error-free environment, the outcome achieved by a wireless scheduler should be identical to that of an equivalent wireline scheduler; (2) the amount of capacity loss suffered by a flow should not be proportional to its bandwidth or its error rate, but should be configurable through administrative controls; (3) it must be possible to administratively bound the amount of capacity that is lost due to location-dependent errors; (4) in the absence of information to the contrary, flows experiencing equal error rates should experience the same capacity loss; and (5) capacity unused by one flow should be distributed "fairly" among other flows. While several of our principles (1, 4, and 5) are similar to those used in other projects [7], [8], the principles dealing with capacity loss (2 and 3) are unique. These two principles will guide our design of a new scheduler model that will allow intuitive control over the effect of capacity loss.

The scheduler model we propose focuses on the insight that, in a wireless environment, we must distinguish between "effort" (air time spent on a flow) and "outcome" (actual useful throughput achieved by the flow). While effort equals outcome in a wireline environment, they can be substantially different in a wireless environment. Imagine that a wireline scheduler evaluates a set of flow requirements and determines that a certain flow would be satisfied by 10% of the error-free link capacity. If that flow belongs to a station experiencing a 50% packet loss rate, spending 10% of the link's *effort* on that flow will result in it achieving *outcome* equivalent to 5% of the link's error-free throughput. But if this flow is critically important to some application, it might make more sense to spend 20% of the link's effort on the flow so that it would achieve its expected outcome, 10% of the error-free link throughput. To address this effort-outcome disconnection, we propose an "effort-limited fair" (ELF) scheduling approach. An ELF scheduler strives to achieve the outcome that is envisioned by users (either a specific throughput for reserved flows or a specific fraction of residual link capacity for best-effort flows), subject to limits on the effort spent on each flow using a per-flow *power factor* setting. The power factor is a control knob that can be used to administratively implement a variety of fairness and efficiency policies.

The remainder of the paper is organized as follows. In the next section we elaborate on the scheduling challenges posed by wireless errors, focusing on providing supporting arguments for principles 2 and 3. In Section III we describe the *power factor* administrative control which defines how a scheduler should respond to capacity loss and in Section IV place our work in the context of related wireless packet scheduling efforts. Section V describes a scheduler implementing the power factor. In Section VI we examine the behavior of this algorithm via trace-driven simulation, and Section VII reports on experience with our prototype implementation.

II. Challenges and Principles

In this section we will analyze the challenges posed by a wireless error environment in order to guide our design process. The insight we gain from this analysis will clarify and support our second and third design principles.

A. Network model

We assume that a wireless network consists of one or more cells and that the bandwidth in each cell is managed by a centralized packet scheduler. One example of a such a cell is an IEEE 802.11 LAN [10] in which a base station is configured to manage all bandwidth via the Point Control Facility.

We assume that the scheduler views traffic as a set of flows. Flows can be individual, e.g., a single TCP connection, or aggregates, e.g., all traffic to a specific host. Throughout the paper we will use a traditional weighted-fair queueing (WFQ) scheduler as an example. In a WFQ scheduler, link time is distributed over the flows according to a set of weights. A WFQ scheduler can be used to implement throughput reservations; in that case, the weights will be adjusted as flows enter and leave the network so that reserved flows maintain their reserved rates.

Finally, we mentioned in the introduction that link errors result in both data loss and capacity loss. Data loss can be addressed using a range of local and end-to-end techniques that affect primarily which data traverses the link and are largely orthogonal to how we deal with capacity loss. In the remainder of this paper we will for simplicity assume that a local error recovery mechanism is used, although this is not a fundamental assumption.

B. Scheduling implications of link error patterns

Wireless networks experience errors on different time granularities, and network designers have devised a variety of error control mechanisms to battle these errors. Error bursts at the bit level are typically addressed by a mixture of error coding and interleaving [9], [5], [6] which reduces but does not eliminate link capacity variation. At a coarser time granularity, some packet errors can be predicted based on packet burst error patterns, and a wireless link scheduler may attempt to reduce the overall packet error rate by swapping transmission slots between stations believed to be experiencing a multi-packet error burst and stations believed to be currently error-free [7], [8]. Again, this approach does not eliminate all capacity loss, because of the imperfect predictability of error bursts and because some interference sources may affect multiple stations or even the base station. We conclude that *capacity loss is a fundamental reality for wireless schedulers*.

C. Reacting to link capacity loss

To discuss the impact of variable link capacity we will use an example of a wireless cell with a capacity of 800 kilobits per second employing a WFQ scheduler to serve two guaranteed flows and two best-effort flows. We will first assume that all errors are location independent, so that all flows experience the same error rate. The flow properties and weights are summarized in Table I. We will use the term *fidelity* to denote the degree to which a scheduler is "faithful" to a particular flow, namely the percentage of the flow's expected throughput on an error-free link that it actually achieves.

TABLE I
Client example

Client Flow	Target rate (kilobits/second)	WFQ Weight
Audio	reserved 8	1.0
Video	reserved 350	44.0
FTP1	available	27.5
FTP2	available	27.5

Let us first consider what happens if 50% of all packets are lost. Since all flows are experiencing errors, most wireless fair queueing schedulers would assign transmission slots to the flows according to their weights in Table I. With this *effort-fair* approach, each flow receives 100% of its expected *effort*, but this is degraded to yield only 50% of its expected *outcome* (Table II), so both rate-sensitive flows fail to achieve their desired rates.

TABLE II
Possible results of a 50% packet error rate

Client	Expected rate (kbit/sec)	Effort-fair rate (kbit/sec)	Preferable rate (kbit/sec)
Audio	8	4 $\otimes$	8 $\surd$
Video	350	175 $\otimes$	350 $\surd$
FTP1	221	110 $\approx$	21 $\approx$
FTP2	221	110 $\approx$	21 $\approx$

Results marked with "$\surd$" represent flows meeting their needs; "$\otimes$" represents un-met needs; "$\approx$" represents flows without specific requirements.

An alternative would be a priority-based scheduler which would allot enough extra effort to the audio and video flows that they would achieve their target rates at the expense of reducing the effort spent on the best-effort FTP flows. This would produce the result shown in the last column of Table II which, given the target rates shown in Table I, is the correct outcome.

TABLE III
LOCATION-DEPENDENT ERRORS

Flow	Error rate (%)	Effort rate (kbit/sec)	Priority rate (kbit/sec)	Outcome rate (kbit/sec)	Desirable rate (kbit/sec)
Video1	0%	100 $\surd$	100 $\surd$	67 $\otimes$	100 $\surd$
FTP1	0%	300 $\approx$	250 $\approx$	200 $\approx$	167 $\approx$
Video2	50%	50 $\otimes$	100 $\surd$	67 $\otimes$	100 $\surd$
FTP2	50%	150 $\approx$	125 $\approx$	200 $\approx$	167 $\approx$
Throughput		600	575	534	534
Efficiency		75%	72%	67%	67%

Results marked with "$\surd$" represent flows meeting their needs; "$\otimes$" represents un-met needs; "$\approx$" represents flows without specific requirements.

While the priority-based scheduler achieves high fidelity in the above scenario, it is too simplistic to be practical. For example, with a 50% error rate, the priority-based scheduler allocates about 89% of the useful bandwidth to reserved flows ($358kbs/400kbs$). Many network managers would argue that this is unfair to the best effort-flows. Things get even worse when the error rate increases. As soon as the error rate is more than 55%, the FTP flows will receive no throughput even though both the audio and video flows will fail to meet their reservations. Since no flow will be satisfied, this is an unreasonable use of the bandwidth. A better allocation would be to have the audio stream meets its reservation while the other three flows split the remaining bandwidth.

Each of these approaches performs well at one extreme of the error rate spectrum and fails at the other extreme. The effort-fair scheduler ensures that a reserved flow experiencing *any* errors will fail to meet its reservation, even if repairing the errors would require only a trivial amount of unfairness to some other flow; the priority-based scheduler performs unacceptably if *any* high-priority flow experiences a high error rate. An important insight of this example is that, while we *should* help the video flow more than the FTP flows because of its importance, we *can* help the audio flow more than any of the others: multiplying its modest throughput needs by even a large factor to compensate for a high error rate won't drive the link to starvation. This sort of decision calls for a control mechanism which admission control can use to map flow rate and importance into a procedure for distributing link capacity loss among flows. In summary, *capacity loss should result in fidelity loss according to administrative controls.*

D. Location-dependent errors

Let us now consider the impact of location-dependent errors using an example with two stations, one of which is error-free while the other experiences a 50% error rate. Each station owns one "thumbnail" compressed video flow (100 kbit/sec) and one FTP flow (best-effort). Table III summarizes the results for four possible schedulers.

With effort-based scheduling, the lost capacity is distributed proportional to the error rate experienced by each flow, i.e., only the second station will suffer capacity loss, and its video flow will not meet its throughput expectation. As we saw above, the effort-based scheduler ensures that every reserved flow experiencing errors fails to meet its reservation.

A priority-based scheduler could ensure that both reserved flows are satisfied. We would then need to define how it should divide the remaining link capacity among the (equal-priority) best-effort flows; Table III assumes this division would be effort-fair. Once again, we cannot actually deploy a priority-based scheduler, since a single high-priority flow experiencing serious errors would starve every flow. Also, the throughput discrepancy between the two FTP flows is very large, and a case could be made that it would be more fair for them to receive the equal throughput.

To address the best-effort fairness issue, we could employ an "outcome-fair" scheduler, which would allocate effort to flows in such a way that all flows would achieve the same percentage of their expected throughput. In this case, an outcome-fair scheduler would ensure that every flow achieves 67% of the throughput it would achieve on an error-free link. In essence, the air time is divided equally among packets sent to the error-free station, lost packets sent to the error-prone station, and packets that successfully reach the error-prone station; each station receives equal throughput and 33% of the air time is wasted. While this would improve fairness among the FTP flows, *neither* video flow would achieve its desired throughput. The outcome-fair scheduler, like the priority scheduler, behaves poorly when any flow experiences a high error rate.

Finally, we will consider a more desirable outcome, in which both video flows meet their needs and both FTP flows receive equal throughput. While this outcome could be accomplished by a hybrid priority/outcome scheduler, this scheduler would become unusable when confronted with any high-error-rate flow.

Notice that the presence of location-dependent errors adds an efficiency dimension to the space of policy options. If all stations experience the same error rate, moving air time from one flow to another does not effect the useful throughput of the cell. However, with different error rates, moving air time from low-error flow to a high-error flow reduces the useful throughput of the cell. This is clearly illustrated by the efficiency row in Table III: outcome-based scheduling results in markedly lower efficiency than effort-based scheduling. All of these considerations call for a scheduler that employs outcome fairness for best-effort flows when error rates are low, uses priority to support reserved flows as long as error rates are moderate, and falls back to effort fairness in the face of very high error rates so that the link will still provide some useful throughput.

This example illustrates two important concepts: *Sometimes flows with different error rates should experience the same fidelity,* but meanwhile *it must be possible to control cell-wide efficiency in the presence of station-dependent errors by administratively bounding the amount of link effort devoted to "fairness."*

III. THE POWER FACTOR

In this section we will present a model of how a wireless link scheduler should adjust flow weights in response to errors in order to create a hybrid between effort fairness and outcome fairness which is parameterized by a single administrative control, the "power factor." We will state a simple weight adjustment criterion, describe the throughput flows achieve as a function of their error rates, show how this weight adjustment can apply to both constant-rate and best-effort flows, and discuss the applicability of this approach.

A. *Weighted fair queueing with adjustable weights*

We will begin with a weighted fair queueing (WFQ) scheduler that distributes effort (air time) according to weights provided by an admission control module. The scheduler will adjust each flow's weight in response to the error rate of that flow, up to a maximum weight defined by that flow's power factor, also provided by the admission control module (see Figure 1).

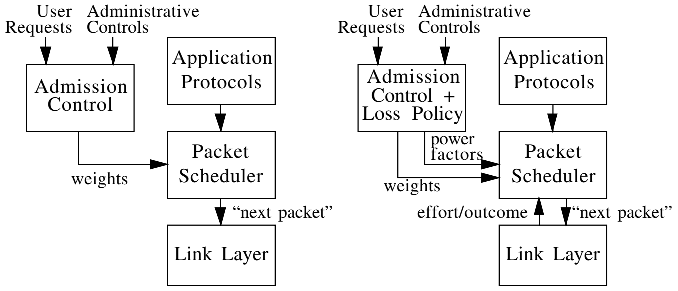

Fig. 1. Scheduler models

For example, a power factor of 200% indicates that a flow's weight should be doubled in a high error environment, which means that it would gain weight relative to flows with a lower power factor, but lose weight relative to flows with a higher power factor. This makes it possible to, for example, increase the link share of voice and web traffic relative to video traffic in a high error environment. We call this type of scheduler *effort-limited fair* (ELF) because it manipulates flow weights to achieve *outcome fairness* subject to a limit on each flow's *effort*.

B. *The power factor*

In order to characterize the behavior of the ELF scheduler across the entire spectrum of error rates, we introduce the following notation. Let us assume we have N flows sharing a link with (error-free) bandwidth B. Each flow has a weight W_i, a power factor P_i, and experiences an error rate E_i. We can now define the adjusted weight of flow i as

$$A_i = min(\frac{W_i}{1-E_i}, P_i \times W_i) \qquad (1)$$

That is, for low error rates we scale the weight W_i to make up for the link errors, but we limit the adjustment to a factor P_i. The crossover point is at error rate $E_i^c = \frac{P_i - 1}{P_i}$. We will refer to the error range $E_i < E_i^c$ as the outcome region and the error range $E_i > E_i^c$ as the effort region. The throughput T_i for flow i is given by the product of the transmission time it receives and its success rate,

$$T_i = (\frac{A_i}{\sum_j A_j} \times B) \times (1 - E_i)$$

To justify this approach, we will look at the behavior of the scheduler under some specific conditions.

First, in an error-free environment ($E_i = 0, \forall i$), the scheduler is equivalent to a traditional WFQ scheduler with weights W_i.

As long as a flow is in its outcome region, A_i adjusts to exactly cancel the flow's reduced success rate $(1 - E_i)$, yielding a throughput of

$$T_i = \frac{W_i}{\sum_j (A_j)} \times B$$

That is, the effective weight of the flow is corrected back to W_i, although that is relative to the adjusted weights of the whole link. If all flows are in their outcome regions and they all experience the same error rate E the throughput of flow i becomes

$$T_i = \frac{W_i}{\sum_j \frac{W_j}{1-E_j}} \times B = \frac{W_i}{\sum_j W_j} \times (B \times (1 - E))$$

Thus the scheduler is equivalent to a WFQ scheduler with the original weights W_i running on a E-degraded link, which is exactly outcome-fair.

At the other end of the spectrum, if all flows are in their effort regions, i.e., $E_i > E_i^c, \forall i$, the throughput becomes

$$T_i = \frac{P_i \times W_i}{\sum_j (P_j \times W_j)} \times (B \times (1 - E_i))$$

This means that the scheduler distributes transmission time to the flows in WFQ fashion and the scheduler is "effort fair" (with adjusted weights).

Finally, one of the motivations for introducing the ELF scheduler approach was to limit how much effort (transmission time) is given to any specific flow, so that one flow experiencing very high error rates cannot degrade the performance of the entire link. The highest fraction of the link time that flow i can take is given by

$$\frac{P_i \times W_i}{(P_i \times W_i) + \sum_{j \neq i} (W_j)}$$

when flow i is in its effort region and all other flows are error-free.

C. Fixed-rate reservations

Providing absolute bandwidth reservations (as opposed to link shares), which WFQ can do, requires additional support in ELF. The reason is that a error-adjusted fraction of a deflated link will be smaller than the expected fraction of the error-free link. We will obtain absolute bandwidth reservations by both adjusting the weights of guaranteed flows upward as described above and simultaneously *reducing* the weights of best-effort flows in a straightforward way.

To support throughput guarantees, we will define G to be the set of guaranteed flows and B_i to be the bandwidth allocated to each flow i in G. Next, we will use fractions of the error-free link as weights, i.e., $W_i = \frac{B_i}{B}, \forall i \in G$. Admission control will be responsible for ensuring that the link is not overcommitted in both the error-free case ($\sum_{i \in G} W_i \leq 1$) and when all guaranteed flows are error-limited ($B_{G_{max}} = \sum_{i \in G} W_i \times P_i \leq 1$). Next, we aggregate all best-effort flows into one virtual flow with a special weight-adjustment function

$$A_{BE} = 1 - \sum_{i \in G} A_i$$

which ensures that the best-effort aggregate flow will consume only whatever link time is left over after every best-effort flow has either achieved its outcome or has reached its crossover error rate E_i^c.

For any guaranteed flow with $E_i < E_i^c$, $A_i = \frac{B_i/B}{1-E_i}$, so its expected throughput

$$T_i = \frac{A_i}{\sum A_j} \times B \times (1 - E_i)$$

becomes the correct value,

$$T_i = \frac{\frac{B_i/B}{1-E_i}}{1} \times B \times (1 - E_i) = B_i$$

The best-effort flows will avoid starvation if $B_{G_{max}} < B$, in which case an error-dependent amount of transmission time will be allocated to the best-effort aggregate flow, which will distribute it among the best-effort flows using exactly the approach of Section III-B.

D. Example

This scheduler can support a variety of policies. For example, the hybrid outcome/effort scheduler described in Table III can be implemented by setting the power factors of each flow to at least 200%. In general, setting a flow's power factor to 100% will cause it to be scheduled in an effort-fair fashion, and raising its power factor will cause it to experience outcome fairness over a wider range of error rates. In particular, it is feasible, albeit probably undesirable, to obtain pure outcome fairness for all best-effort flows by setting their power factors to infinity.

E. Choosing power factors

So far we have assumed that an admission control module can set appropriate per-flow power factors though we have not specified how they might be chosen. One possibility would be to adapt an existing wireline admission control module in a straightforward fashion. If the link error rate is expected to rarely exceed a certain critical value E^c, assign every admitted reserved flow a power factor of $\frac{1}{1-E^c}$ and stop admitting new reserved flows when they occupy $1 - E^c$ of the error-free link rate, which is when their worst-case air-time requirements would consume the entire link. Another possibility would be to assign power factors according to flow classes. For example, flows requesting 8kb/s or 64kb/s could be categorized as voice flows and assigned a power factor of 300%; a second throughput range, appropriate for compressed video, could be assigned a power factor of 150%; and a value could be chosen for all best-effort flows. Such an admission control module would need to avoid overcommitting the link, and would need some policy to determine how much of the link could be assigned to each class.

F. Discussion

The proposed "power factor" scheduler model meets the requirements outlined in the introduction. By setting the power factor appropriately, administrators can control the degree to which the fidelity of a flow will be maintained in the presence of errors. Selection of the power factor should consider the relative importance and demands of flows (e.g., audio is typically more valuable than video while requiring less bandwidth), and should also consider fairness issues across classes (e.g., reserved traffic should not be able to starve best-effort traffic). The power factor can also be used to control efficiency. For example, by keeping all power factors below 200%, we can keep the efficiency over 50% (for error rates under 50%).

In this section we demonstrated how adding a per-flow power factor setting transforms a wireline WFQ scheduler into a hybrid effort/outcome WFQ scheduler. We believe that the power factor is also a useful abstraction for other schedulers. One example is a priority scheduler. We could associate power factors with the different priority levels to avoid high priority traffic completely starving low priority traffic or flows suffering from location-dependent errors consuming a disproportionate fraction of the link capacity in a futile attempt to obtain their expectations. Schedulers which offer more complex service characterizations will require additional work. A deadline-based scheduler or any scheduler attempting to meet delay or jitter guarantees will clearly need to track outcomes for individual packets rather than entire flows.

In the next section we will compare the ELF model to other proposed wireless link scheduling models before proceeding to describe and evaluate our scheduler implementation in the remainder of the paper.

IV. Related Work

Fair queueing in a error-prone wireless environment is examined in [7]. The WPS wireless packet scheduler extends weighted round robin in an attempt to provide fairness, swap time allocations between stations experiencing error bursts and currently error-free stations, avoid scheduling bursts (so a packet burst won't collide with an error burst), avoid polling stations experiencing an error burst, and ensure that stations not experiencing errors receive their expected throughputs. WPS takes into account imperfect information about link state and client queue occupancy. A recent extension of this work [11] adds a variety of attractive properties, such as separate management of delay and bandwidth and graceful trading off of bandwidth between leading and lagging flows. Evaluation is via a simulator using a Markov-model error environment. The main difference between this work and ELF is our belief that it is frequently appropriate for error-free flows to yield throughput to flows experiencing errors.

A set of formal fairness properties for evaluating wireless versions of wired packet fair queueing algorithms is proposed in [8]. Among these properties are that error-free stations should not lose service to error-prone stations and that stations which receive extra throughput due to another station's unavailability are not forced to pay back this excess via a long service outage. These properties are embodied in a proposed algorithm, Channel-condition Independent packet Fair Queueing (CIF-Q), that achieves these fairness properties in the face of errors. Both analytical and simulation results are presented, the latter based on an on-off error model. ELF differs by being able to recover from the errors of some flows at the expense of other flows.

The *Server-Based Fairness Approach* [12] creates one or more virtual "server" flows that are used to compensate flows for errors they have experienced in the past. The amount and timing of compensation depend on the amount of capacity reserved for each flow's compensation server, the relative weight of that flow compared to others sharing the same compensation server, and the error rates experienced by all flows compensated by that server. This approach is powerful due to its generality. For example, it is possible for SBFA embedded in HFS-C to approximate ELF by constructing a tree with two top-level nodes (reserved and best-effort, with the best-effort node allocated a tiny fraction of the link), pairing every ELF flow with its own compensation server flow, and assigning weights appropriately. One difference between the resultant scheduler and ELF is that ELF will try to meet the latency expectations of low-rate flows experiencing errors at the expense of slightly increasing the latencies of high-rate flows, even those not experiencing errors. Also, this SBFA construction might suffer from efficiency issues because the compensation server flows (50% of all flows) would frequently transition between being idle and active. The ELF scheduler is designed to react efficiently to link errors, treating them as the common case. We believe that the very generality of SBFA calls for a simple and intuitive fairness model such as our power-factor approach.

Utility-fair bandwidth allocation [13] presents a framework for allowing flows to specify how much damage they incur as a result of varying amounts of throughput reduction. The scheduler then allocates throughput to each flow so that all flows perceive the same subjective quality. While this approach expresses more information about a flow's needs than ELF does, it is unclear how to apply it to location-dependent errors: it would appear that if a single flow experiences a 100% error rate then all flows will experience a quality level of zero.

We believe that an explicit model of the desired outcome of a scheduling algorithm in the face of errors, as provided by WPS, CIF-Q, and SBFA, is valuable. However, we believe it is important in certain situations to give guaranteed flows "special treatment" when they encounter errors, that it is often attractive to slightly reduce overall link efficiency to obtain some amount of outcome fairness among best-effort flows, and that it must meanwhile be possible to protect the link against flows with very high error rates. ELF provides a way of coping with the fundamental challenges posed by wireless link capacity loss.

V. Scheduler Algorithm

A. Scheduler model

In this section we will present an algorithm for a particular ELF scheduler based on weighted round-robin (WRR). This scheduler will support reserved flows with absolute-rate reservations and best-effort flows which distribute the remaining capacity via weights. Flows in both classes will be bounded by effort limits. We use packet weighted round-robin (WRR) instead of weighted fair queueing, measuring throughput in packet slots. One reason for using WRR instead of WFQ is simplicity. Also, wireless networks are often slot-based for synchronization and power-management purposes. Finally, the prototype network we use for experiments in (Section VII) has a relatively high per-packet cost, so charging on a per-packet instead of a per-byte basis is actually quite reasonable.

B. Tracking Deserved Throughput, Effort, and Outcome

Protecting a link against excessive consumption by a subset of flows is an interesting challenge. The most straightforward approach would be to measure the error rate of each flow and of the link and then to calculate the air time each flow should get based on the definition of the power factor (Section III). However, this assumes that a link scheduler can accurately measure a flow's error rate and the current capacity of a wireless link. These measurements are not straightforward. For example, determining the error rate of a low-throughput flow is difficult, since observations are infrequent; single-packet observations may fail to detect long outage bursts, so increasing the flow's share of the link could increase that flow's error rate. Determining the available link capacity is even more difficult, considering the presence of location-dependent errors. The problem here is that transmitting on behalf of different flows yields different link-capacity measurements. If a single flow has a 50% error rate and a weight of 50%, and is allocated enough effort to achieve its expected *outcome*, it will consume the entire link, so link capacity will be 50%. If it is allocated only its expected 50% of the link's *effort*, and the remainder of the link is given to error-free flows, link capacity will be 75%. These two considerations argue for a scheduling algorithm that achieves the desired link allocation without relying on explicit computations of per-flow error rates or whole-link capacity.

C. Algorithm Overview

We will explain our scheduling algorithm using the pseudo-code in Figures 2 and 3. The code embodies several simplifying assumptions. All flows have data queued at all times. Transmission time is slotted into fixed-size packets, though an adaptive link layer may choose to send shorter packets to avoid interference-related truncation. Each packet transmission fully succeeds or fails (there are no sub-packet acknowledgements).

Admission control converts each flow's weight into its expected inter-transmission interval (a flow expecting 50% of the link would transmit every two time slots). The transmission intervals of best-effort flows apply not to actual link time but to a virtual time that increases only when no guaranteed flow is eligible to transmit. Within each flow group (guaranteed or best-effort), the scheduler operates the same way: each time a flow's transmission interval passes, `tick` allocates it both throughput and an appropriate amount of effort; it is then eligible for transmission until it either achieves that throughput or exhausts that effort. When multiple flows are eligible for transmission, `choose` selects the most urgent. We will choose the flow whose outcome is currently lagging its deserved throughput by the greatest percentage, breaking ties by serving low-throughput flows first.

```
/* per flow state kept by scheduler:                      */
/*  inter - inter-transmission interval, 1/rate          */
/*  last  - timestamp of last eligibility to send        */
/*  power - power factor (as percentage)                 */
/*  deserve - amount of *outcome* flow deserves          */
/*  effort  - amount of *effort* flow is entitled to */

/* "tick": update the deserve and effort of each flow  */
tick(flow-list, now)
{
   foreach flow (flow-list)
      if (flow.last + flow.inter <= now)
         shouldserve(flow, now);
}

shouldserve(flow, now)
{
   ++flow.deserve;
   flow.effort += flow.power / 100;
   cap_effort(flow);
   flow.last = now;
}

/* "choose" next flow from a class of flows */
choose(flows, &resultflow)
{
   int bestwlag = 0;  /* best weighted lag */
   int bestinter = 0;
   int bestdeserve = 0;
   flow bestflow = NULL;

   foreach flow (flows) {
      /* weighted lag: deserve / rate */
      int wlag = flow.deserve * flow.inter;

      /* must have effort */
      if (flow.effort < 1)
         continue;

      /* most deserving flow has highest % lag */
      /* tie breakers: low-rate flows first;   */
      /* then flows with more absolute lag    */

      if ((wlag < bestwlag) ||
         ((wlag == bestwlag) && (flow.inter > bestinter)) ||
         ((wlag == bestwlag) && (flow.inter == bestinter)
            && (flow.deserve < bestdeserve)) {
         continue; /* not the most deserving */
      }
      /* becomes the most deserving */
      bestwlag = wlag;
      bestinter = flow.inter;
      bestdeserve = flow.deserve;
      bestflow = flow;
   }
   *resultflow = bestflow;
}
```

Fig. 2. Pseudo-code for core of the scheduler

The main scheduling function `schedule`, is called by the link layer when it is ready to send a packet (Figure 3). It will first try to select an eligible guaranteed flow before moving on to best-effort flows. That is, guaranteed flows are served until each is either satisfied with its outcome or limited by expended effort, and the best-effort flows share the leftover bandwidth. The main scheduler decision procedure operates as follows. First it uses `tick` to inform guaranteed flows of the passage of time, and then it asks `choose` whether any guaranteed flow should currently transmit. If not, it moves on to the best-effort class, calling `tick` with the best-effort virtual time and then invoking `choose`. While reserved time increases for each link time slot, best-effort time increases only when no best-effort flow is eligible, as a consequence of the fact that the best-effort virtual flow is almost always overcommitted. If no guaranteed flow is eligible and there are no best-effort flows, the scheduler will slightly advance the schedule of the next-eligible guaranteed flow.

```
/* "schedule" picks next flow to be served */
schedule(&selectedflow)
{
    static int now, dilatednow;
    flow flow;

    ++now;
    tick(protected-flows, now);
    choose(protected-flows, &flow);
    if (!flow && length (unprotected-flows) > 0) {
        choose(unprotected-flows, &flow);
        while (!flow) {
            ++dilatednow;
            tick(unprotected-flows, dilatednow);
            choose(unprotected-flows, &flow);
        }
    }
    if (!flow} {
        /* advance schedule */
        nextflow(protected-flows, &flow);
        shouldserve(flow, now);
    }
    *selectedflow = flow;
}

update(flow,outcome) /* call-back from link layer */
{
    --flow.effort;
    if (outcome == SUCCESS) {
        --flow.deserve;
        cap_effort(flow);
    }
}

cap_effort(flow) /* limit accumulated effort  */
{
    /* retain no more than 4 packets' effort */
    int slack = 4;
    int cap = ((flow.deserve/100)+slack)*flow.power;
    flow.effort = max(flow.effort, cap);
}

/* "nextflow" picks the flow that will fire next */
nextflow(flows, &flow)
{
    flow nextflow;
    int nextfire = INFINITY;

    foreach flow (flows)
        int fire = flow.last + flow.inter;

        if (fire < nextfire)
            nextflow = flow;
            nextfire = fire;
    *flow = nextflow;
}
```

Fig. 3. Pseudo-code for core of the scheduler - part 2

At this time the link layer transmits a packet for the designated flow, employing whatever combination of packet length and error coding recently history suggests is prudent for successful communication with that flow's peer machine. Before the next scheduling decision is made, the link layer will inform us of the transmission outcome using the function `update`, which will bill the designated flow for the effort expended and update its achievement value appropriately. This assumes the existence of a fast link-level acknowledgement, a feature common to many wireless LAN MAC protocols [10], [14], [15].

Finally, whenever we update the status of a flow (functions `tick` and `update`) we check whether the stored effort exceeds a threshold using the function `cap_effort`. This is so that a flow cannot accumulate a large effort bank while its error rate is low and seize the link for an extended period when it suddenly increases. We believe `cap_effort` contains a plausible heuristic.

D. Possible extensions

A question ignored so far is to what extent best-effort flows should experience "fairness" over a long term. That is, if a flow experiences errors for a period and loses throughput as a result, is it entitled to regain that lost throughput later? This corresponds to the question of whether there is a cap on each flow's `deserve` value. If this value is allowed to grow without bound, a flow will eventually reclaim the lost throughput, at the expense of other flows in its class, when it experiences a low-error period. Alternatively, this value could be capped or aged.

A related issue is what should happen to a flow that becomes idle. One possible policy is that its `deserve` and `effort` would cease to increase. Instead, we could allow an idle flow to accumulate both deserve and effort up to some threshold; this would then allow the flow to burst for a short while when it becomes active. Note that deciding whether a flow is idle may be difficult when error rates are high.

This scheduler implementation is orthogonal to swapping in the sense that, if there is some good predictor of when a station is unreachable, `choose` can easily skip known-unreachable stations.

This scheduling approach could be used as the policy for an 802.11 Point Control Function (PCF). If we assume that one station is in a position to discover the outcomes of most packet transmissions, the LAN could operate according to the more-efficient Distributed Control Function most of the time, and the controlling station could then use the PCF period to allocate extra effort to stations according to their power factors.

VI. SIMULATION

To evaluate our scheduling algorithm in a repeatable fashion we subjected it to a simple trace-based simulation. The simulator assumes all flows are continuously busy, and records throughput allocation decisions made by the scheduler (ignoring how real transport protocols might react to allocation variations). The link layer used in the simulation is based on earlier work [5], and can reclaim substantial capacity even in the face of challenging error patterns. As we discussed in Section II, this is largely orthogonal to the question of how we distribute the variable link capacity across the flows, which is the focus of this paper.

We will plot the error-free link-level throughput each flow achieves, as a percentage of what it would expect in the absence of errors, on its own independent vertical axis. Each plot will include a "Link" pseudo-flow which represents the total link throughput as a fraction of the error-free link throughput. Each flow will be represented by a moving-window average throughput, sampled at the same rate as its expected inter-transmission interval (that is, a flow expecting 50% of the link is sampled five times as frequently as a flow expecting 10%) and each point on the graph is the mean of the 10 most recent samples.

We will present three simulator runs. In the first, we evaluate the scheduler's performance in the motivating scenario of Table II. Two protected flows, audio and video, expect 1% and 44% of the link, while two unprotected flows will share the remainder equally. The audio flow has a power factor of 300% in an attempt to ensure it will survive most plausible link errors. The video flow has a power factor of 223%, a value chosen so that it and the audio flow will consume the entire link in periods when the error rate is 50% or more. The two best-effort flows each have a power factor of 120%, which will enable outcome-based fairness between them in the face of light errors.

Throughout the trace, the link will experience a uniformly distributed 50% packet loss rate which is independent of which flow's packet is being transmitted. We would like the reserved audio and video flows to experience little or no disruption while the two best-effort flows should equally divide the remainder of the link. The results of this simulator run appear in Figure 4 and are what we would hope to observe.

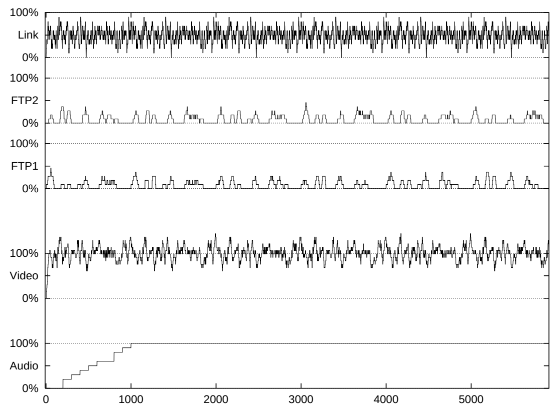

Fig. 4. Plot of a straightforward 50% loss scenario suggested by Table II

.

Figure 5 displays how the same flows would fare in a more interesting error environment. The error trace [9] includes both packet losses and bit corruptions, varying in severity due to a person moving through the main signal path. The sudden sharp throughput losses leave the audio flow essentially unharmed. Since the video flow requires much more throughput, it cannot escape all injury, but the scheduler insures it is quickly compensated. Each link throughput drop immediately affects both best-effort flows.

Figure 6 is an example of a location-dependent error pattern: the video flow and one of the best-effort flows belong to a station experiencing significant errors, while the other two flows belong to a station encountering none. The error pattern is a smoothly

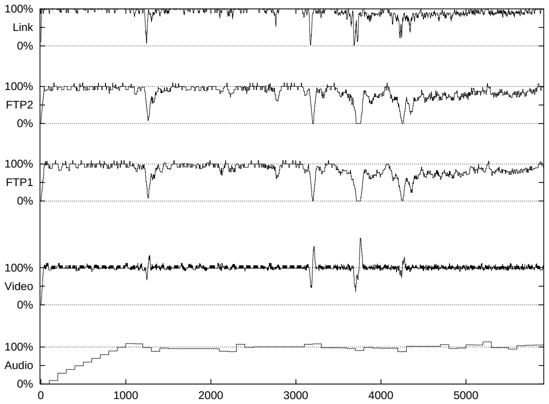

Fig. 5. Response to a dynamic real-world error trace.

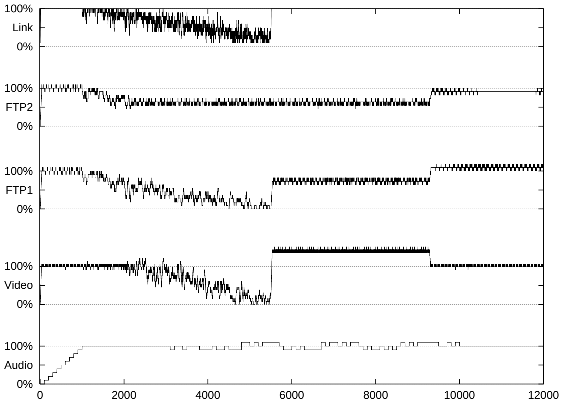

Fig. 6. Location-dependent error trace.

increasing frequency of packet loss, ranging from 10% to 90%, surrounded by periods without losses. The video flow's power factor has been lowered to a more-realistic 140%. As the losses begin, the two best-effort flows share the burden equally, though only "FTP1" experiences actual transmission errors. Soon, however, flow "FTP1" exhausts its outcome region and begins to lose throughput relative to "FTP2." Later, link errors begin to overwhelm the video flow's ability to demand extra link time. From this point on, the burden of any increase in the error rate is borne by these two flows. When the link no longer experiences errors, we observe two interesting phenomena. First, the video flow receives more than its expected share of the link so it can clear its lag, but its power factor still limits its link time so that the FTP flows are not starved. Second, the ill-fated "FTP1" flow receives more throughput than "FTP2" so it can achieve long-term fairness with "FTP2." Again, this short-term unfairness is limited by its power factor.

These scenarios illustrate the main goals of our scheduler. We have observed prioritization of reserved flows over best-effort flows coupled with effort-limited outcome-based fairness. The scheduler performs intuitively across a wide range of desired flow throughputs (1% to 44% of the link) and error rates (10% to 90%).

VII. Experimental Prototype

In addition to our simulator-based evaluation, we wished to verify that our scheduler would function acceptably in a more "real-world" environment. We inserted the ELF scheduler into our existing prototype wireless LAN, built from Intel 80486 and Pentium laptops running NetBSD 1.2 and using 915 MHz PCMCIA card WaveLAN [16] units. The kernel device driver includes a simplified poll/response Medium Access Control (MAC) protocol, similar in spirit to the IEEE 802.11 Point Control Function [10]. In our experiments, one laptop operates as a master/base station, while the others use the slave protocol. We will report user-level throughput numbers obtained by the standard kernel TCP stack. In order for TCP to make meaningful progress in the face of these high error rates, we employ a transparent link-level error control mechanism [9]. As we discussed earlier, error control is largely orthogonal to scheduling, so we could have obtained similar experimental results by disabling error control and reporting UDP throughput observed by the receiver instead of TCP throughput observed by the sender.

The scheduler consists of 400 lines of code, of which approximately 150 lines are the heart of the algorithm and the remainder is glue code, memory management, tracing support, and trace-replay error injection code.

We will present `tcptrace` output for four TCP streams generated by a master and two slave stations. Each slave receives one reserved flow (link fraction: 5%, power factor: 300%) and one best-effort flow (power factor: 150%). One slave station experiences no errors and the other begins by experiencing no errors but then experiences 10 seconds each of 20% and 50% packet loss rates before returning to an error-free condition.

We would expect that the reserved flows always achieve their expected throughputs and that the best-effort flows would experience equal throughput whenever their error rates were below 30%. The traces in Figure 7 meet our expectations (the initial upward spike in throughput displayed for each flow is an artifact of the throughput averaging done by `tcptrace`). The slight sag in throughput observed by the reserved flows is an artifact of the software implementation of our MAC protocol (we are forced to use conservative timeouts to avoid packet collisions, so a lost packet takes slightly more air time than a packet that arrives). The prototype implementation confirms the more detailed simulator results presented earlier.

VIII. Conclusion

To effectively address wireless errors which may be severe, time-varying, and location-dependent, we propose an "effort-limited fair" (ELF) scheduling approach which extends the scheduler with an explicit mechanism for controlling its behavior in the presence of capacity loss. An ELF scheduler strives to achieve the outcome that is envisioned by users (e.g., weighted link sharing or fixed-rate reservations) while limiting the effort spent on a flow using a per-flow parameter called the power factor, which can be used to administratively implement a variety of fairness and efficiency policies.

We implemented an ELF WRR scheduler that supports reservations and that uses the power factor to limit the per-flow effort applied to combat errors. Simulation clearly depicts ELF scheduling in action. We also implemented the scheduler in a NetBSD operating system kernel, and show that it achieves appropriate division of user-level TCP throughput under various error conditions.

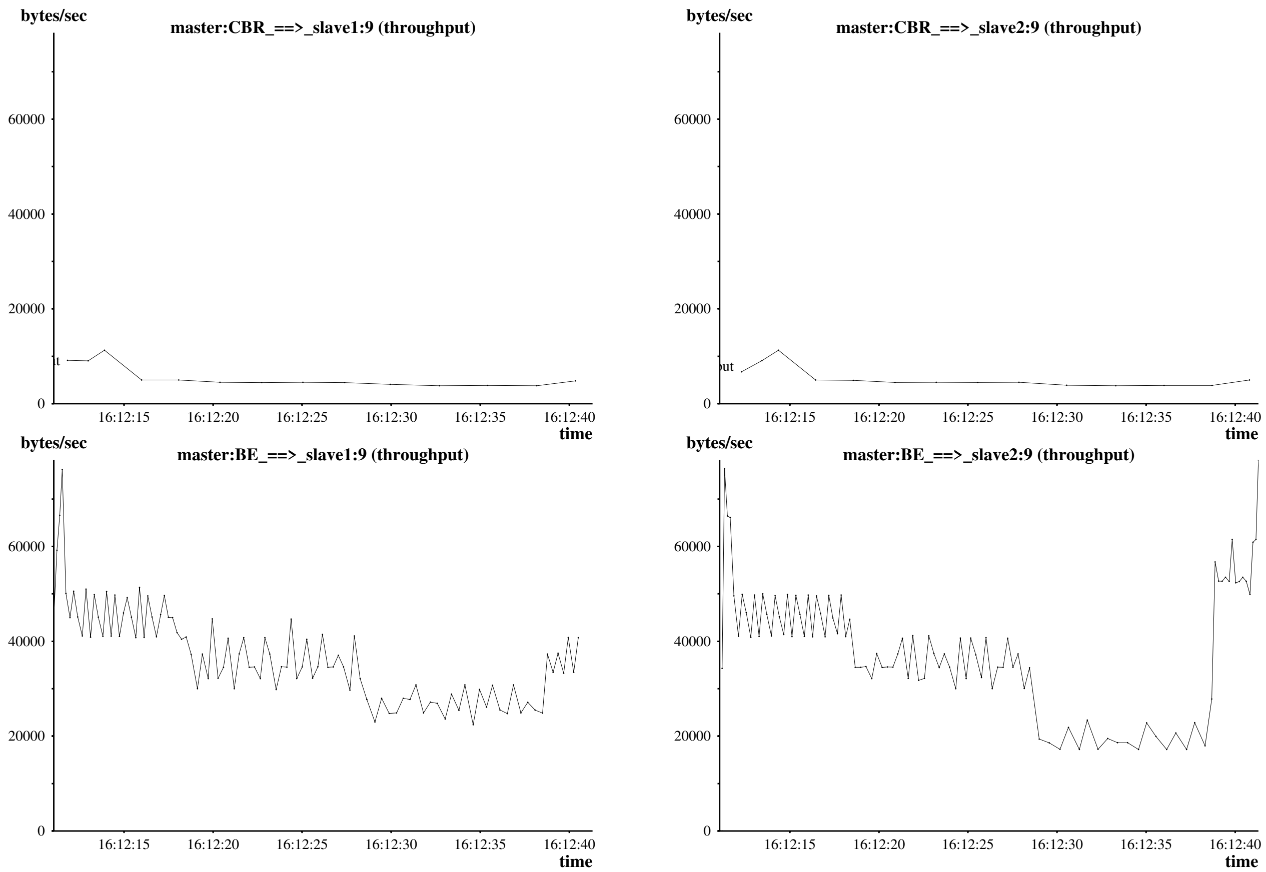

Fig. 7. Traces of reserved and best-effort TCP streams

ACKNOWLEDGEMENTS

The WaveLAN driver we modified was written by Bob Baron of the Coda research project. Hui Zhang and Ion Stoica provided extensive comments on our scheduling proposals. A reviewer comment led to a noticeable improvement in the introductory material.

This research was supported in part by the Defense Advanced Research Project Agency/ITO monitored by NRaD under contract N66001-96-C-8528.

REFERENCES

[1] A. Bakre and B. R. Badrinath, "I-TCP: Indirect TCP for mobile hosts," in *Proceedings of the 15th International Conference on Distributed Computing Systems*, May 1995, pp. 136–143.

[2] Raj Yavatkar and Namrata Bhagawat, "Improving end-to-end performance of TCP over mobile internetworks," in *Mobile '94 Workshop on Mobile Computing Systems and Applications*, December 1994.

[3] Hari Balakrishnan, Venkata N. Padmanabhan, Srinivasan Seshan, and Randy H. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," *IEEE/ACM Transactions on Networking*, December 1997.

[4] Antonio DeSimone, Mooi Choo Chuah, and On-Ching Yue, "Throughput performance of transport-layer protocols over wireless LANs," in *Proceedings of IEEE GLOBECOM 1993*, December 1993, pp. 542–549.

[5] David Eckhardt and Peter Steenkiste, "Improving Wireless LAN Performance via Adaptive Local Error Control," in *Sixth International Conference on Network Protocols*, Austin, TX, October 1998, IEEE Computer Society.

[6] Paul Lettieri and Mani B. Srivastava, "Adaptive frame length control for improving wireless link throughput, range, and energy efficiency," in *Proceedings of IEEE INFOCOM '98*, San Francisco, CA, March 1998, pp. 564–571.

[7] Songwu Lu, Vaduvur Bharghavan, and Rayadurgam Srikant, "Fair scheduling in wireless packet networks," in *Proceedings of ACM SIGCOMM '97*. September 1997, IEEE Computer Society.

[8] T. S. Eugene Ng, Ion Stoica, and Hui Zhang, "Packet fair queueing algorithms for wireless networks with location-dependent errors," in *Proceedings of INFOCOMM '98*. IEEE Communication Society.

[9] David Eckhardt and Peter Steenkiste, "A Trace-based Evaluation of Adaptive Error Correction for a Wireless Local Area Network," *Mobile Networks and Applications (MONET)*, 1999, Special Issue on Adaptive Mobile Networking and Computing.

[10] IEEE Local and Metropolitan Area Network Standards Committee, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Std 802.11-1997, The Institute of Electrical and Electronics Engineers, New York, New York, 1997.

[11] Songwu Lu, Thyagarajan Nandagopal, and Vaduvur Bharghavan, "A wireless fair service algorithm for packet cellular networks," in *Proceedings of The Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM '98)*, Dallas, TX, October 1998, ACM SIGMOBILE.

[12] Parameswaran Ramanathan and Prathima Agrawal, "Adapting Packet Fair Queueing Algorithms to Wireless Networks," in *Proceedings of The Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM '98)*, Dallas, TX, October 1998, ACM SIGMOBILE.

[13] Guiseppe Bianchi, Andrew T. Campbell, and Raymond R.-F. Liao, "On Utility-Fair Adaptive Services in Wireless Networks," in *Proceedings of the Sixth International Workshop on Quality of Services (IWQOS '98)*, Napa Valley, CA, May 1998, IEEE Communications Society.

[14] Phil Karn, "MACA–a new channel access method for packet radio," in *Proceedings of the 9th ARRL/CRRL Amateur Radio Computer Networking Conference*, September 1992.

[15] Vaduvur Bharghavan, Alan Demers, Scott Shenker, and Lixia Zhang, "MACAW: A media access protocol for wireless LANs," in *ACM SIGCOMM '94*, August 1994, pp. 212–225.

[16] Bruce Tuch, "Development of WaveLAN, an ISM band wireless LAN," *AT&T Technical Journal*, pp. 27–37, July/August 1993.