ERROR-AWARE SCHEDULING AND ITS EFFECT ON EFFICIENCY AND FAIRNESS

Pablo Serrano, David Larrabeiti, Manuel Urueña
Universidad Carlos III de Madrid
Departamento de Ingeniería Telemática
Av. Universidad 30, E-28911 Leganés, Madrid, Spain
{pablo,dlarra,muruenya}@it.uc3m.es

Antonio G. Marques
Departamento de Ciencias de la Comunicación
Universidad Rey Juan Carlos
Cl Tulipán s/n E-28933 Mostoles, Madrid, Spain
antonio.garcia.marques@urjc.es

Abstract This paper describes a mechanism to adapt an existing wireline scheduling algorithm for a WLAN Access Point, by taking into account the error ratio affecting each flow. This enhancement is based on the idea of weighting flows according to their error ratio. Users connected over error-prone channels get their bandwidth share increased, up to a point where the overall efficiency breaks down, and the mechanism is reverted. The cost of this mechanism in terms of fairness is also addressed.

Keywords: Efficiency, Fair Queuing, WLAN.

Introduction

In wireless networks, time and location-dependent signal attenuation, interference, fading and noise result in a different error ratio for each flow sharing a packet switched link. This has motivated an intense research activity in wireless scheduling in the last years.

Wireless scheduling addresses the problem of how to provide a weighted fair allocation of bandwidth even under changing channel conditions (an excellent review on this topic is given in [Bharghavan et al., 1999|). These and more recent proposals (see [Raghunathan et al., 2002, Liu et al., 2003, Wong et al., 2003, Wang and Chin, 2001|) are based on the capability to probe the channel before transmission: flows that do not receive the corresponding bandwidth share *now* will receive it *later*. This assumption is not applicable to WLAN, where no probing mechanism is available.

We propose to adapt an existing conventional fair scheduling mechanism [Zhang, 1995] for WLAN, by taking into account the error ratio

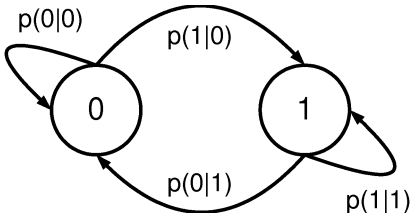

Figure 1. Gilbert Cell

Table 1. Gilbert Cell parameters

Flow	$p(0\|0)$	$p(1\|1)$
1, 2	0.1	0.9
3, 4	0.538	0.846
5	0.9	0.1

affecting each flow. This error ratio estimation serves as a basis for the development of two complementary mechanisms: flow compensation and throttling. Compensation aims to increase fairness, whereas throttling tries to keep up an acceptable overall efficiency.

One main problem of compensation is that it may yield to very low goodputs, if applied unrestrictedly. Unlike previous works, compensation is not only bounded in terms of *allocated rate* but also in the *amount of time* a flow is granted a preferential treatment. Furthermore we introduce throttling of flows suffering from high loss channels in an attempt to prop up efficiency, and determine its effect on fairness.

In WLAN-specific scenarios, available algorithms |Vaidya et al., 2000, Banchs and Perez, 200| deal with distributed QoS provisioning by modifying 802.11b |IEEE, 1999| behavior. Our work considers a centralized scenario, dealing with the Access Point (AP) operation in presence of Mobile Nodes (MN).

The rest of this paper is organized as follows. The target scenario is introduced in Section 1. Section 2 describes the proposed scheduler extension. Simulation results and conclusions are given in Sections 3 and 4.

1. Scenario

We focus on a 802.11b scenario, where no prioritized MNs with different channel conditions share a common medium in order to transmit from and to an AP. We consider the transmission of packets from the AP to the MNs. Each MN is associated with one and only one flow identifier at the AP[1]. We aim to provide a fair allocation to all these users, but taking into account channel efficiency.

We make three assumptions about this scenario:

- Retransmission mechanisms are not implemented. Whether lower or upper layers should deal with retransmission is a passionate

Table 2. Variables and parameters of the algorithm

Term	*Definition*
N_F	Number of flows
$bytesErr_i$	Bytes transmitted with error since activation of compensation
$\hat{\rho}_i$	Mean packet probability error since activation of compensation
ω_{comp}	Weight reserved for compensation
ω_i	Weight given to flow i
ϵ_i	Number of consecutive errors of flow i
Ω	Threshold for maximum compensation
θ_i	Number of rounds for a flow i to be skipped
Θ	Maximum θ_i allowed for all i

discussion (see [Saltzer et al., 1984] for a classical review of the topic). However, if retransmissions need to be implemented, it would be straightforward: if a packet needs n retransmissions prior to its receipt, it would be considered (from the point of view of the scheduler) as $n-1$ transmissions with error and 1 successful transmission.

- Collisions are not taken into account, as we are only interested in packet errors due to bad channel conditions.
- Only one transmission rate is available (although with multiple rates available fair *temporal access* to the medium could be provided, instead of fair bandwidth allocation).

The channel is modeled as a two-state Markov Model (or *Gilbert Cell*, Fig. 1), following the empirical characterization of [Khayam and Radha, 2003]. The '0' state means that a packet gets lost, and the '1' state means that the packet gets through. The $p(i|j)$ is the probability of transition from state j to state i. The error probability is then: $P(0) = p(0|1)/(p(0|1) + p(1|0))$

2. The Algorithm

We can build our algorithm on top of almost any available wireline scheduling algorithm, by enhancing their functionality via a *configuration interface*. A new entity, called EAS (error-aware scheduler), modifies the wireline scheduler by sending a weight vector ($\{\omega_i\}$) and a throttling vector ($\{\theta_i\}$) (see Table 2 for a summary of notation). These vectors, sent from the EAS to the scheduler, support the implementation of compensation and throttling, respectively:

- The *compensation* is implemented by dynamically assigning weights to flows. Flows that need compensation are given higher weights (during some packets) than error-free flows.
- On the other hand, θ_i represents the number of turns to be lost by a packet from flow i: assume flow i has a Head of Line packet

of size L_i, and $\theta_i > 0$. When the next flow to dequeue is i, the scheduler computes L_i as work given to i and decrements θ_i by one. Only when $\theta_i = 0$ the packet is actually dequeued.

The EAS re-configures the Scheduler in a real-time fashion, by processing the received ACK (or its absence) after each transmission. Then it performs the *compensation/throttling* calculation. Our scheme generalizes easily to any scheduling algorithm, by implementing the described configuration interface (the work in [Ramanathan and Agrawal, 1998] also enjoys this feature).

Compensation

In order to define a compensation mechanism, we have decided to limit the maximum fraction of bandwidth (or weight, ω_{comp}) available for compensation (following the philosophy of [Ramanathan and Agrawal, 1998]).

If the compensation mechanism is triggered for a given flow, this flow has always a minimum weight guaranteed, given by $\omega_{i0} = (1-\omega_{comp})/N_F$ (we impose $\sum_{i=1}^{N_F} \omega_i + \omega_{comp} = 1$). Compensation is proportional to lost bytes, and it vanishes with subsequent successful transmission. Flows with a high number of consecutive errors ($\epsilon_i > \Omega$) are not given any compensation; i.e. we consider them *irrecoverable errors* which will damage goodput of less error-prone channels. This way, the compensation mechanism is given by :

Give flow i the weigth ω_i, according to

$$\omega_i := \begin{cases} \omega_{i0} + \Delta\omega_i, & \epsilon_i \leq \Omega \\ \omega_{i0}, & \epsilon_i > \Omega \end{cases}$$

where

$$\Delta\omega_i := \omega_{comp} \times \underbrace{\frac{bytesErr_i}{\sum_{i=1}^{N_F} bytesErr_i}}_{\omega_{comp}\ partition} \times \underbrace{\hat{\rho}_i}_{damping}$$

Both the counter $bytesErr_i$ and the mean packet error probability ($\hat{\rho}_i$) start to measure from the first packet loss. If ϵ_i surpasses Ω, the compensation stops and they are both reinitialized to zero. Also, when $\Delta\omega_i/\omega_i \leq 0.1$, the compensation is finished (and counters are reinitialized). This way, not only the rate of compensation is bounded (via ω_{comp}), but also the amount of compensation (implicitly): either a flow perceives a low number of errors (and then $\hat{\rho}_i$ imposes the reduction of $\Delta\omega_i$), or it perceives a great number of errors (and thus $\epsilon_i > \Omega$ and compensation is deactivated).

Throttling

Starting from a number of consecutive errors ($\epsilon_i > \Omega$), the compensation mechanism will not give any increment of weight to flow i. But

even in these situations, flows with very ill-behaved links will keep on wasting the radiolink during an error burst. In this case, we propose not only to give not any compensation, but also to *throttle* these flows in an adaptive manner (the more consecutive errors acquired, the more rounds a flow will be passed by if there is any are other flow in the system).

We start to throttle from the $\Omega + 1$-th consecutive packet error, in order to detect a series of packet losses. In order to avoid starvation, Θ limits the maximum number of rounds a packed can be passed by. Then, θ_i is given by:

$$\theta_i := \begin{cases} 0, & \epsilon_i \leq \Omega \\ \epsilon_i - \Omega, & \Omega < \epsilon_i \leq \Theta + \Omega \\ \Theta, & \epsilon_i > \Theta + \Omega \end{cases}$$

It should be taken into account that throttling *passes by* a flow. But if no other flow is competing for the link, the algorithm will be unnoticeable (although being active). Hence, in over-provisioned links no throttling will take place: only in scenarios where flows compete aggressively for the link the algorithm will really favor well-behaved channels.

3. Simulation Results

The simulations were carried out with OMNeT++ [OMNeT++, 2003], a discrete-event simulator. With $N_F = 5$, we have five sources and five MNs, connected to the AP via five Gilbert Cells. We defined three kinds of channels associated to these MNs (see Table 1):

- Channels with little probability of errors (high quality links), modeling MNs close to the AP: Flows 1, 2.
- Channels whose model was taken from measurements of [Khayam and Radha, 2003] (average quality links): Flows 3, 4.
- Channels very error-prone (low quality links): Flow 5.

We implemented two MNs associated to the first channel type, two of the second type, and one of the third type (later on we will change the proportion of channel types, but not the number of MNs).

Transmission rate for the wireless link is 2 Mbps. The scheduling algorithm is SCFQ ([Golestani, 1994]). Traffic interarrival is exponential, and packet length is uniformly distributed between 1000 and 1500 bytes. All MNs are sent the same amount of traffic, considering three distinct cases: 282.84 kbps, 400 kbps and 564 kbps, for an aggregate of 1.41 Mbps, 2 Mbps and 2.82 Mbps respectively. This way we can analyze the performance in under and oversubscribed environments, covering a 3 dB range. The simulation run procedure was implemented following the two sequential method of [Nakayama, 1994]. A minimum of $m = 10$ batch means were collected for a 90% of confidence interval of $\epsilon = 0.1$ relative size (with a previous warm-up period, sized 5 batch means).

First the simulation results for the compensation algorithm alone are presented (Section 3), which achieves greater fairness at the expense

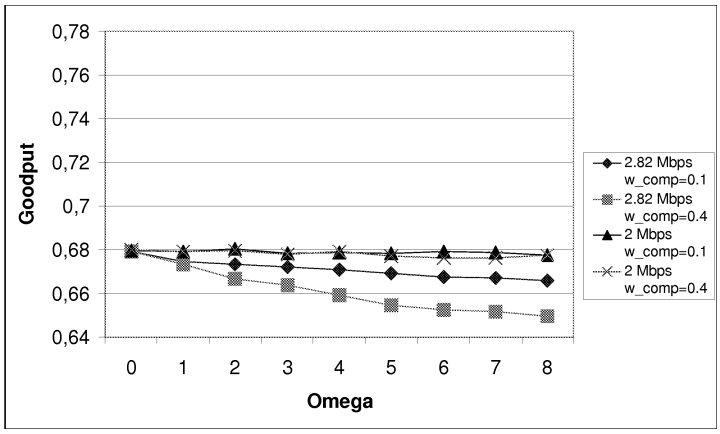

Figure 2. Goodput vs Ω, compensation only

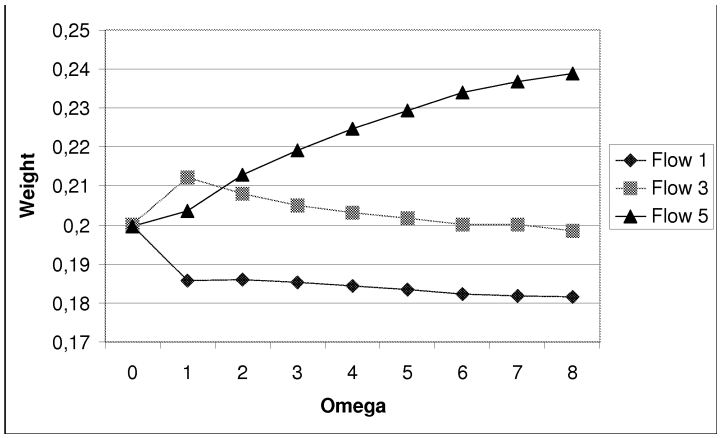

Figure 3. ω_i vs Ω, compensation only

of degrading system overall efficiency. In Section 3 the counterpart of compensation is shown: throttling, which improves system performance by reducing bandwidth of flows with errors. Finally, in Section 5 the performance of the complete algorithm is presented.

Compensation-only Algorithm

First we are going to define the efficiency or goodput[2] as the ratio $bytesAcked/totalBytesTransmitted$. Figure 2 shows the obtained system goodput vs Ω for two transmission rates and two values of ω_{comp} (being $\Omega = 0$ the case when no algorithm is active). In an oversubscribed scenario (2.82 Mbps of incoming traffic) and with $\omega_{comp} = 0.4$, goodput reduces noticeably with the maximum number of consecutive

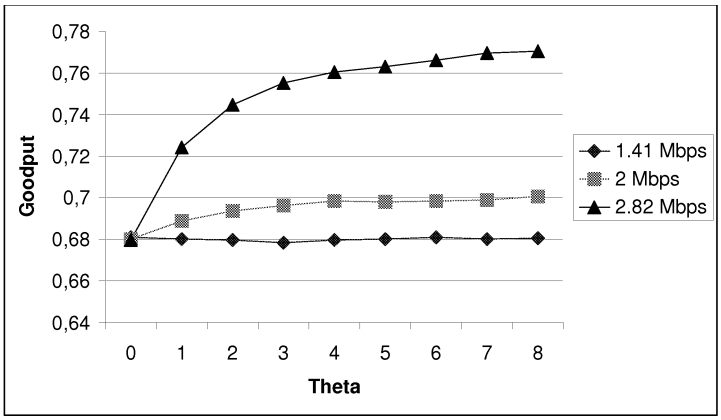

Figure 4. Goodput vs Θ, throttling only

errors allowed, Ω. The explanation of this behavior is straightforward: the algorithm is giving flows with error-prone channels more and more bandwidth (up to 40% of channel capacity), and thus the proportion of error-free frames is diminishing ($\Delta\omega_i$ is assigned proportional to the number of bytes with errors). With $\omega_{comp} = 0.1$ the diminishing on goodput is less noticeable, since it bounds the amount of channel given to compensation. On the other hand, when incoming traffic is 2 Mbps, the compensation mechanism is not noticeable, because flows are not always competing for the medium.

Figure 3 shows the mean ω_i obtained for different values of the maximum number of consecutive errors allowed (for $\omega_{comp} = 0.4$ and 2.82 Mbps). The flow with the greatest error probability (fifth flow) obtains a 25% of improvement over its nominal weigth, just because almost every packet transmitted to it is lost. And even with such an effort, this flow will only perceive a tiny improvement on its particular goodput. Thus, compensation by its own might lead to resource squandering. This issue is addressed by throttling.

Throttling-only Algorithm

In this case we will show the throttling performance with $\Omega = 0$, so every packet error is punished. Thus, by incrementing the maximum throttling allowed (via Θ) we are improving system performance (see Fig. 4), because the more a flow perceives consecutive errors, the more rounds it will be skipped (and error-free channels will monopolize the system). But throttling is adaptive in nature, as it has been discussed on Section 2: only when flows compete aggressively for the medium (2.82 Mbps) its behavior is noticeable. Thus, in undersubscribed scenarios (1.41 Mbps), despite all rounds a packet should be passed by due to errors, if no other flow is competing for the channel, it will be transmitted.

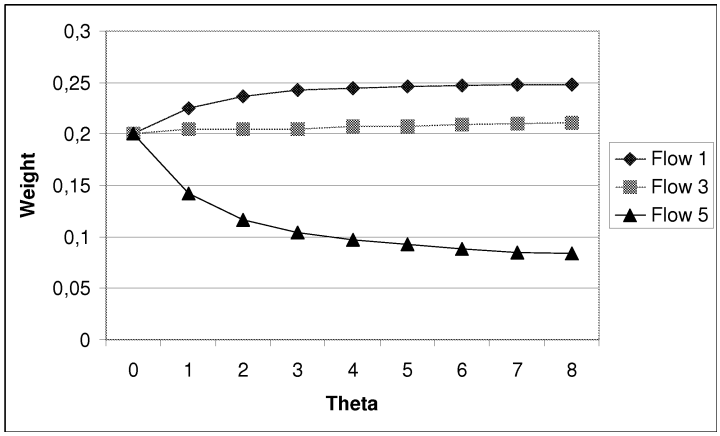

Figure 5. ω_i vs Θ, throttling only

Table 3. Probability of n or more consecutive errors

Flow	1	2	3	4
1,2	10^{-1}	10^{-2}	10^{-3}	10^{-4}
3,4	0.25	0.1345	**0.0724**	0.0389
5	0.9	0.81	0.729	0.656

By looking at the mean values of ω_i (Fig. 5), it is clear that the increase of goodput comes from the decrease of ω_5, the weight assigned to the flow with the worst channel. Compensation mechanism guarantees a minimum fraction of bandwidth to all flows (ω_{i0}), but if the throttling mechanism is triggered this value will be lowered (the maximum number of throttling can be fixed via Θ: the minimum ω_i at any moment is $\omega_{i0}/(\Theta + 1)$).

Complete Algorithm

In order to analyze the performance of the complete algorithm, first we are going to give values to Ω and Θ (instead of performing a sweeping on all possible values). Even with $\Theta = 1$, bandwidth of flows with error-prone channels is reduced to a maximum of half of its original value, while allowing a noticeable improvement on goodput (Fig. 4).

On the other hand, Ω aims to distinguish between flows with recoverable errors, and flows associated to an error-prone channel. Table 3 shows the cumulative probabilities for n or more errors of all flows. Second row values are taken from the empirical characterization (|Khayam and Radha, 2003|), and thus we choose them so as to perform the discrimination. This way, with $\Omega = 2$, the probability for a flow of the second kind to be throttled is less than 10% (see the highlighted value), which is a reasonable threshold.

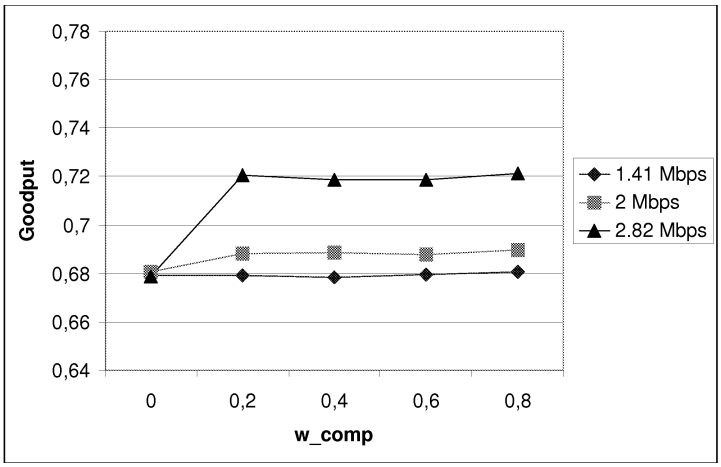

Figure 6. Efficiency, complete algorithm

Figure 6 shows the system goodput for four values of ω_{comp}, being again $\omega_{comp} = 0$ the case where the mechanism is inactive. The independence between goodput and ω_{comp} comes from the fact that only throttling provokes the goodput rise, while compensation aims to support a fair allocation of resources. Again, only in an oversubscribed scenario (2.82 Mbps) the improvement is noticeable, due to the adaptive nature of throttling.

The system increases overall efficiency. In order to measure properly the impact on fairness, a quantity is needed. Based on the concept of proportional fairness [Kelly, 1997], the following measure is defined:

$$Fairness = \sum_{i=1}^{N_F} log\left(\frac{bytesACK_i}{Simulation\ Time}\right) \tag{3}$$

This is the *cost* paid for the efficiency increase: if the mechanism aims to improve goodput by throttling aggressively flows with error-prone links, this punishment will be taken into account in Eq. 3 (preventing flow starvation).

For the $\omega_{comp} = 0.6$ case, the efficiency rises from 0.68 to 0.72, while fairness (calculated via Eq. 3) varies from 51.0 to 50.9. Thus we have almost a 6% of increment on efficiency, while the reduction on fairness is just 0.2%. Thus, less error frames appear in the system, although it keeps on providing almost the same proportional fairness.

Different Distribution of Channel Conditions

In order to analyze the performance of the algorithm in different scenarios, the proportion of channel types is modified from the original one (described in Table 1). Instead of two users with a *high quality* chan-

Table 4. Goodput Improvement for Different Distributions of Channel Types

	N_{AQ}					
N_{LQ}	0	1	2	3	4	5
0	0.00	-0.01	-0.01	-0.01	-0.01	-0.01
1	0.07	0.07	0.07	0.08	0.08	
2	0.13	0.13	0.14	0.14		
3	0.17	0.17	0.17			
4	0.16	**0.15**				
5	-0.01					

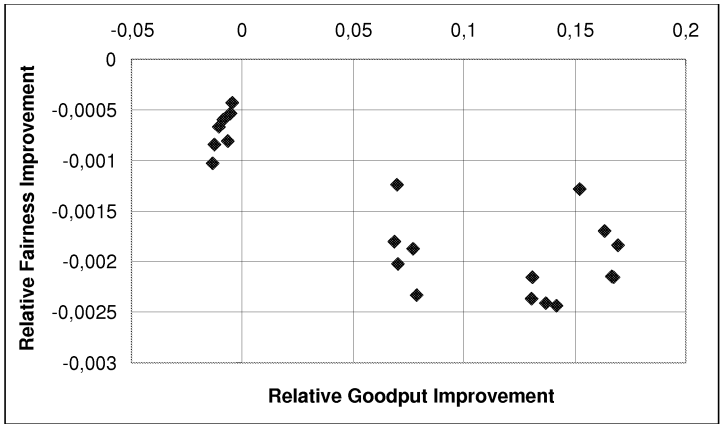

Figure 7. Goodput improvement vs Fairness improvement for different distributions of channel types

nel (HQ), two with a *average quality* channel (AQ) and one with a *low quality* channel (LQ), all possible combinations of $N_F = 5$ flows were simulated. Then, both the improvement introduced by the algorithm and its effect on system fairness were measured, and compared to the situation where the mechanism was inactive.

Results for efficiency are shown on Table 4, where the number of AQ channels increases to the right, and the number of LQ channels increases downwards (the number of HQ channels is implicit, $N_{HQ}+N_{AQ}+N_{LQ} = 5$). For example, the 0.15 value (highlighted) corresponds to a scenario with no MN with a HQ channel, 1 MN with an AQ channel and 4 MNs with LQ channels. It is evident that goodput improvement rises with the number of LQ channels. When all channels are equal, the goodput remains almost the same. There is an average relative improvement on efficiency of 8%.

On the other hand, the algorithm provokes a 0.16% average relative decrease of fairness. Figure 7 shows the achieved trade off between fairness and goodput, for all considered scenarios. The relative decrease of fairness is almost unnoticeable (at most, 0.25%), while the improvement on goodput is never less than 5%, when there is room for improvement.

4. Conclusions

In WLAN environments the absence of a channel probing mechanism prevents direct application of most wireless fair scheduling algorithms. Moreover, even if they were applicable, the compensation mechanisms implemented may provoke resource squandering, because a flow may deserve endless compensation.

In this work we have presented a mechanism to adapt wireline scheduling algorithms to WLAN Access Points, by implementing a bounded (in terms of bandwidth and time) compensation mechanism, and a throttling mechanism. This novel type of double-bounded compensation, and the application of throttling to flows associated with error-prone channels, has been shown to perform well on scenarios with different distributions of link quality.

Acknowledgments

This work has been partly supported by the European Union under the e-Photon/ONe Project (FP6-001933) and by the Spanish Research Action CICYT CAPITAL (MEC, TEC2004-05622-C04-03/TCM). We also thank the reviewers of this paper for their valuable comments.

References

|Banchs and Perez, 200| Banchs, A. and Perez, X. (200). Distributed fair queuing in ieee 802.11 wireless lan. In *IEEE International Conference on Communications (ICC 2002), New York, April 2002.*

|Bharghavan et al., 1999| Bharghavan, V., Lu, Songwu, and Nandagopal, T. (1999). Fair queuing in wireless networks: issues and approaches. In *Personal Communications, IEEE, Vol.6, Iss.1, Feb 1999 Pages:44-53.*

|Golestani, 1994| Golestani, S. (1994). A self-clocked fair queueing scheme for broadband applications. In *Proceedings of IEEE INFOCOM 94, pages 636-646, Toronto, CA, June 1994.*

|IEEE, 1999| IEEE (1999). Ieee 802.11b, part ii: Wireless lan medium access control (mac) and physical layer (phy) specifications: Highspeed physical layer extension in the 2.4 ghz band.

|Kelly, 1997| Kelly, F. (1997). Charging and rate control for elastic traffic. *European Transactions on Telecommunications, 8*, pages 33–37.

|Khayam and Radha, 2003| Khayam, Syed A. and Radha, Hayder (2003). Markov-based modeling of wireless local area networks. In *Proceedings of the 6th international workshop on Modeling analysis and simulation of wireless and mobile systems*, pages 100–107. ACM Press.

|Liu et al., 2003| Liu, Yonghe, Gruhl, S., and Knightly, E.W. (2003). Wcfq: an opportunistic wireless scheduler with statistical fairness bounds. In *Wireless Communications, IEEE Transactions on, Vol.2, Iss.5, Sept. 2003 Pages: 1017- 1028.*

|Nakayama, 1994| Nakayama, M. (1994). Two-stage stopping procedures based on standardized time series. In *Management Science 40, 1189–1206.*

|OMNeT++, 2003| OMNeT++ (2003).

|Raghunathan et al., 2002| Raghunathan, Vijay, Ganeriwal, Saurabh, Schurgers, Curt, and Srivastava, Mani B. (2002). E2wfq: An energy efficient fair scheduling policy for wireless systems. In *International Symposium on Low Power Electronics and Design (ISLPED'02), Monterey, CA, pp. 30-35, August 12-14, 2002.*

[Ramanathan and Agrawal, 1998] Ramanathan, Parameswaran and Agrawal, Prathima (1998). Adapting packet fair queueing algorithms to wireless networks. In *Mobile Computing and Networking*, pages 1–9.

|Saltzer et al., 1984| Saltzer, J. H., Reed, D. P., and Clark, D. D. (1984). End-to-end arguments in system design. *ACM Trans. Comput. Syst.*, 2(4):277–288.

|Vaidya et al., 2000| Vaidya, Nitin H., Bahl, Paramvir, and Gupta, Seema (2000). Distributed fair scheduling in a wireless LAN. In *Mobile Computing and Networking*, pages 167–178.

[Wang and Chin, 2001] Wang, Kuochen and Chin, Yi-Lon (2001). A fair scheduling algorithm with adaptive compensation in wireless networks. In *Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE, Vol.6, Iss., 2001 Pages:3543-3547 vol.6.*

[Wong et al., 2003] Wong, W.K., Zhu, Haiying, and Leung, V.C.M. (2003). Soft qos provisioning using the token bank fair queuing scheduling algorithm. In *Wireless Communications, IEEE [see also IEEE Personal Communications], Vol.10, Iss.3, June 2003 Pages: 8- 16.*

[Zhang, 1995] Zhang, H. (1995). Service disciplines for guaranteed performance service in packet-switching networks. In *Proc. IEEE, vol. 83, Oct 1995, pp. 1374-96.*