A Wireless Fair Service Algorithm For Packet Cellular Networks

Songwu Lu Thyagarajan Nandagopal Vaduvur Bharghavan
Coordinated Science Laboratory
University of Illinois
email: {slu, thyagu, bharghav}@timely.crhc.uiuc.edu

Abstract

In order to support diverse communication-intensive real-time and non real-time data flows over a scarce, varying and shared wireless channel with location-dependent and bursty errors, we define a service model that has the following characteristics: *short-term fairness* among flows which perceive a clean channel, *worst-case delay bounds* for packets, *short-term throughput bounds* for flows with clean channels and *long-term throughput bounds* for all flows with bounded channel error, *optimal schedulable region*, and support for both *delay sensitive and error sensitive* data flows.

We present a wireless fair service algorithm, and show that it achieves the requirements of the service model through both analysis and simulation. The key aspects of the algorithm are the following: (a) an enhanced fair queueing based service scheme that supports decoupling of delay and bandwidth, (b) graceful service compensation for lagging flows and graceful service degradation for leading flows, (c) support for real-time delay sensitive flows as well as non real-time error sensitive flows, and (d) implementation of the wireless fair service algorithm within the framework of the simple and robust CSMA/CA wireless medium access protocol.

1 Introduction

In recent years, there has been a tremendous growth in the wireless networking industry. With the increasing usage of mobile and wireless networks in both indoor and outdoor environments, the issue of providing fair channel access among multiple contending hosts over a scarce and shared wireless channel has come to the fore. In wireline networks, fair queueing has long been a popular paradigm for providing fairness and bounded delay link access [1, 3]. However, adapting fair queueing to the wireless domain is non-trivial because of the unique problems in wireless channels, such as location-dependent and bursty errors, and channel contention. Consequently, the fair queueing algorithms proposed in literature for wireline networks do not apply directly to wireless networks. In a related work, we proposed a wireless fair queueing algorithm that addressed a number of the issues raised above [8]. Our algorithm was further refined by Ng et al. in [10]. However, a number of theoretical and practical issues in wireless network access were not addressed in either of these works. In this paper, we propose a novel wireless scheduling and channel access algorithm that provides fair channel access with bounded delay, minimum short-term and long-term throughput, delay and bandwidth decoupling with the optimal schedulable region, support for both real-time and non real-time data flows, and that can be implemented in the framework of a simple CSMA/CA wireless channel access protocol.

The key issues in wireless channel access are the following: (a) the *channel capacity is dynamically varying*, (b) *channel errors are location-dependent and bursty* in nature, (c) there is *contention* in the channel among multiple mobile hosts, (d) there are hidden and exposed stations, (e) *mobile hosts do not have global channel state* (in terms of which other hosts contending for the same channel have packets to transmit, etc.), (f) the scheduling must take care of both *uplink and downlink flows*, and (g) mobile hosts are often constrained in terms of processing power and battery power. Thus, any wireless scheduling and channel access algorithm must work within the constraints imposed by the environment. At the same time, in order to support communication-intensive real-time and non real-time data flows over a scarce, varying and shared channel, the service model must have the following characteristics: (a) it must provide *short-term fairness* among flows which perceive a clean channel, (b) it must provide *delay bounds* for packets, (c) it must provide *short-term throughput bounds* for flows with clean channels and *long-term throughput bounds* for all flows with bounded channel error, (d) it must *optimize the schedulable region* by accepting flows with different decoupled delay/bandwidth requirements, and (e) it must support both *delay sensitive and error sensitive* data flows. Our service model seeks to provide channel-conditioned deterministic guarantees with the above five properties for uplink and downlink flows in a cellular environment. We term this service model as the *'wireless fair service model'*.

In this paper, we present a wireless fair service algorithm that achieves the wireless fair service model for packet cellular networks, and briefly analyze its properties. Through simulations, we show that the algorithm performs well for typical wireless communication environments with diverse flow requirements.

The rest of the paper is organized as follows. Section 2 discusses related work and the motivation for this paper. Section 3 presents the wireless fair service algorithm and describes a practical implementation of the scheduling algorithm and medium access protocol. Section 4 evaluates the performance of our algorithm through analyses and simulations. Section 5 concludes the paper.

2 Background and Motivation

In this section, we first present the channel model. We then review related work in wireline and wireless fair queueing.

Finally, we motivate our work and briefly compare our approach to related work.

2.1 Channel Model

We assume a packet cellular architecture in which each cell has a base station. Transmission is in terms of fixed size packets. Our algorithms are applicable for variable size packets as well, though we consider fixed size packets for simplicity, and also because this is acceptable for wireless channels. There is a single channel for both uplink and downlink flows, and for both data and signaling. Each packet transmission involves a RTS-CTS handshake between the mobile host and the base station that precedes the data transmission. Thus, at most one packet transmission can be in progress at any time in a cell. Even though all the mobiles and the base station share the same channel, errors and contention are location dependent. Location-dependent errors happen due to fades, interference, etc. Location-dependent contention may happen in case neighboring cells use the same logical channel for communication (somewhat unusual in cellular networks, but more common in ad hoc networks where hidden and exposed stations are a common occurrence). A flow is said to perceive a clean channel if both the communicating end-points perceive clean channels and the handshake can take place. A flow is said to perceive a dirty channel if either end-point perceives a channel error. We assume a mechanism for the (possibly imperfect) prediction of channel state. This is reasonable, since channel errors are highly correlated between successive slots, every host can listen to the base station, and the base station participates in every data transmission by sending either the Data or the ACK in the RTS-CTS-Data-ACK sequence of our CSMA/CA based medium access protocol.

Errors in the wireless channel typically occur over short bursts and are highly correlated in successive slots, but uncorrelated over long time windows. Thus, errors can be well represented by a 2-state Markov model. In this paper, we do not make any assumptions about the exact error model, though we do assume that the number of errors for a given time window of size T_i is bounded, i.e. we will assume that no more than e_i errors can occur in any time window of size T_i, where e_i and T_i are per-flow parameters for flow i. All the guarantees in the wireless fair service model are 'channel-conditioned', i.e. conditioned on the fact that flow i perceives less than e_i errors in any time window of size T_i.

2.2 Wireline Fair Queueing

In wireline networks, fair queueing has long been a popular paradigm [1, 3] for providing bounded delay access to a shared unidirectional channel, and hence for providing guaranteed quality of service. All fair queueing algorithms are based on the notion of approximating the fluid model, in which packet flows are modeled as fluids that traverse a shared pipe. Consider a set of flows $i \in F$ that share a channel. Let flow i have a weight r_i, where r_i is the number of bits of flow i served in a single 'round' by the fluid fair queueing server. Fluid fair queueing guarantees that for an arbitrary time window $[t_1, t_2]$ during which any two flows i and j are backlogged (i.e. they have bits to transmit), $W_i(t_1, t_2)/r_i = W_j(t_1, t_2)/r_j$, where $W_i(t_1, t_2)$ is the service (in bits) received by flow i in the time window $[t_1, t_2]$. Essentially, fluid fair queueing divides the channel capacity at any instant among backlogged flows in the proportion of their weights. As a direct consequence of this model, flows that do not have any bits to transmit at some time cannot be compensated at a later time. Since networks switch flows at the granularity of packets rather than bits, and since the switching is non-preemptive (i.e. all bits in a packet are transmitted back-to-back), packetized fair queueing algorithms must approximate the fluid model. Thus, the goal of a packetized fair queueing algorithm is to minimize $\mid W_i(t_1, t_2)/r_i - W_j(t_1, t_2)/r_j \mid$ for any two backlogged flows i and j over an arbitrary time window $[t_1, t_2]$. This is achieved by assigning a 'virtual time' start tag and finish tag to each packet, and serving the packet with the minimum finish tag, where the virtual time of the channel corresponds to the current round being served in the corresponding fluid fair queueing model. Thus, the k^{th} packet of flow i, p_i^k, that arrives at time $A(p_i^k)$, is assigned a start tag $S(p_i^k) = \max\{V(A(p_i^k)), F(p_i^{k-1})\}$, and a finish tag $F(p_i^k) = S(p_i^k) + L_i^k/r_i$, where L_i^k is the size (in bits) of packet p_i^k and r_i is the weight of flow i. $V(t)$, the virtual time corresponding to time t, maintains the 'round number' of the fluid fair queueing model at time t. Let $\sum_{i \in B(t)} r_i$ be the bits transmitted in each round, where $B(t)$ is the set of flows that are backlogged at time t. Then, $dV/dt = C/\sum_{i \in B(t)} r_i$, where C is the channel capacity.

2.3 Wireless Fair Queueing

Wireline fair queueing algorithms do not account for wireless channel issues such as location dependent errors and wireless medium access. In order to adapt fair queueing to the wireless domain, we proposed the Idealized Wireless Fair Queueing algorithm (IWFQ) in a related work [8]. Specifically, IWFQ allows for location-dependent and bursty channel error, and for uplink and downlink flows in a shared broadcast channel. IWFQ defines two types of service: the *error-free service*, where all flows perceive a clean channel at all times, and the *real service*, where some flows may perceive dirty channels at any given time. Each flow is labeled as being 'leading', 'lagging', or 'in sync' depending on whether its real service is ahead of, behind, or in accordance with its error-free service. At each time, the packet with the minimum finish tag on a clean channel is transmitted. Thus, flows that experience clean channels can lead their error-free service at the expense of flows that perceive channel errors. Lagging flows are allowed to retain their precedence in terms of finish tags; thus, when a lagging flow perceives a clean channel, its packets will have the highest precedence (i.e. lowest finish tags), and it will be able to catch up on its lag. We bound the amount of lag and lead. Bounding the lag is equivalent to bounding the amount of compensation that a flow can receive due to past channel errors. Bounding the lead is equivalent to bounding the amount of compensation that a flow will pay for having received additional service due to channel error in other flows. Based on the bounds of lag and lead, and the ability to maintain precedence for lagging flows, IWFQ is able to establish channel conditioned guarantees for maximum delay and minimum throughput. While IWFQ is an ideal service model, we proposed the Wireless Packet Scheduling algorithm (WPS) to address the practical issues in wireless scheduling and medium access such as having the base station schedule uplink and downlink flows, propagating information about uplink flows to the base station, medium access, etc.

In a recent work, Ng et. al. [10] proposed a new service model, called Channel-condition Independent Fair (CIF) model, that subsumes four key properties that a packet fair queueing algorithm should possess in a wireless environment: (a)

delay and throughput guarantees for error-free flows, (b) short-term fairness for error-free flows, (c) long-term fairness for error-prone flows, and (d) graceful service degradation for leading flows. They also presented a CIF-Q algorithm that achieves the CIF service model. However, decoupling of delay and bandwidth is not considered in CIF and CIF-Q, and practical MAC layer issues associated with wireless fair queueing are not addressed. Additionally, in sync flows are disturbed and become lagging flows when a lagging flow leaves the set of active flows.

IWFQ has the property that a flow that has perceived an error channel for a long time and then perceives a clean channel can essentially capture the channel for short time periods (till it catches up with the error-free service). Both the WPS algorithm in [8] and the CIF algorithm in [10] address this problem by ensuring that compensation is given more gradually to lagging flows. While WPS provides a weighted round-robin implementation of IWFQ, CIF maintains the fair queueing properties of IWFQ.

In another recent work, Ramanathan et. al. have proposed a simple approach called the Server Based Fairness Approach [12], which is based on the assumption that all flows which perceive clean channel states must receive their promised service and not a fraction of the service as in [8, 10], and flows which have not received adequate service due to channel error are supplemented with additional bandwidth via special sessions called Long-Term Fairness Servers (LTFS). However, they do not explain how SBFA can achieve all the desirable properties for packet fair queueing algorithms as identified in [10], except long term throughput guarantee.

In addition to the algorithms described above [8, 10, 12], there have recently been a number of other wireless channel allocation algorithms that seek to provide fair service [5, 9]. To our knowledge, none of these algorithms either define or seek to provide the types of quality of service parameters that we seek to provide with the wireless fair service model.

2.4 Service Model and Motivation

The basic philosophy of fluid fair queueing is to not compensate for flows that 'miss their turn'. Hence, if a flow does not have a packet to transmit, it cannot later reclaim the missed slot. However, we seek to distinguish between missing service due to no backlog, and missing service due to channel error. If a flow is backlogged but cannot send packets either because it perceives a bad channel or because of contention avoidance, then it should be compensated at a later time.

At the same time, flows that are neither leading nor lagging their error-free service, i.e. flows on error-free channels that did not take advantage of channel errors in other flows to increase their transmission rate, should not be impacted by the compensation model.

In this paper, we have chosen to *compensate a backlogged flow that is unable to transmit a packet during its scheduled slot only if some other flow transmits a packet during this slot.* Thus, our compensation model is the following: if a flow is not backlogged, or if the flow perceives a channel error but no other flow is able to transmit during its scheduled slot, then it does not receive compensation. If a flow perceives a channel error but some other flow can transmit instead, then it will receive compensation later, and the flow that transmitted in its place will relinquish a slot later. Hence, leading flows give up their lead by relinquishing transmission slots in the future and lagging flows make up their lag by transmitting in slots that the leading flows relinquish. The scheduling algorithm thus provides transparency to short location-dependent error bursts by swapping slots between different flows. Flows that neither lead nor lag remain unaffected by our compensation model.

An inherent limitation of fluid fair queueing is that the delay observed by the packets of a flow is tightly coupled with the fraction of the channel given to the flow among all backlogged flows. IWFQ [8] and CIF-Q [10] suffer from this limitation too. Additionally, IWFQ does not provide short term fairness among flows with clean channels; CIF provides short term fairness, but does not account for any of the practical issues in wireless medium access. As described in Section 1, the wireless fair service model accounts for all of the above issues, and the wireless fair service algorithm presented in this paper achieves the goals of the model.

3 Wireless Fair Service Algorithm

The wireless fair service algorithm is composed of five key components that work in concert in order to achieve wireless fair service in a simple and efficient manner:

- The *error-free service model*, that decouples delay and bandwidth requirements and optimizes the schedulability region.
- *Slot queues and packet queues*, which allow us to support both delay sensitive and error sensitive flows in a single framework and also decouple connection-level packet management policies from link-level packet scheduling policies.
- The *lead and lag model* in wireless service, which enables us to determine which flows are leading or lagging their error free service, and by how much.
- The *compensation model*, which enables us to compensate for lagging flows at the expense of leading flows, and thus addresses the key issues of bursty and location-dependent channel error in wireless channel access.
- The *wireless medium access protocol*, which provides a simple, robust, and elegant distributed implementation of the wireless fair service algorithm within the popular CSMA/CA paradigm.

In this section, we describe each of the components of the algorithm, and finally put together the different components. In the next section, we show via theoretical analysis and simulation, that our algorithm achieves wireless fair service.

3.1 The error-free service model

In this section, we describe the basic scheduling model for the wireless fair service algorithm. For simplicity, we first consider an error-free service, i.e. where none of the flows perceive a channel error. In subsequent sections, we will introduce channel error and study its impact on our algorithm.

For the error-free service, we adapt the fluid fair queueing model, but with weights for both rate and delay. Consider a shared channel with a set of flows F, where each flow $i \in F$ has a rate weight of r_i and a delay weight of Φ_i. As in fair queueing, we assign a start tag, $S(p_i^k)$, and a finish tag $F(p_i^k)$, for the k^{th} packet of the flow i according to the following algorithm:

- $S(p_i^k) = \max\{V(A(p_i^k)), S(p_i^{k-1}) + L_i^{k-1}/r_i\}$
  where L_i^k is the length of the k^{th} packet of the flow i, $A(p_i^k)$ is the arrival time of the packet, and $V(t)$ is the virtual time at time t.
- $F(p_i^k) = S(p_i^k) + L_i^k/\Phi_i$
- $dV/dt = C(t)/\sum_{i \in B(t)} r_i$
  where $B(t)$ is the set of backlogged flows at time t and $C(t)$ is the instantaneous channel capacity at time t.
- At each time t, we transmit the packet with the minimum finish tag among those packets whose start tag is not greater than $V(t) + \varrho$, where ϱ is the 'lookahead' system parameter.

Essentially, we have a two-level service model, in which a flow i is drained into the scheduler according to a rate weight r_i, but served according to a delay weight Φ_i. The virtual time moves similar to fluid fair queueing, but each packet is assigned a finish tag that is the sum of its start tag (when it is expected to arrive in virtual time) and its deadline (the maximum number of rounds, in virtual time, that the packet is expected to wait before being served). Essentially, the introduction of Φ_i enables us to switch the schedule among arrived packets in a way that decouples the long term rate of arriving packets (according to r_i) from the delay bounds for packets (according to Φ_i and r_i). The lookahead parameter ϱ provides a measure of the number of packets over which we can locally switch the schedule of packets without disrupting the long-term rate. Basically, a lookahead of ϱ enables the scheduler to 'look ahead in virtual time' by ϱ rounds, and swap the service order of packets that have already arrived in current time with packets that arrive in the future but have shorter deadlines.

The parameter ϱ plays a very interesting role in determining the schedulable region of the algorithm. If $\varrho = 0$, then the error-free weighted fair service algorithm is basically the worst-case fair weighted fair queueing algorithm [4] with deadline-based swapping among the head-of-line packets. If $\varrho = \infty$, then the error-free weighted fair service algorithm is basically the earliest deadline first algorithm, which has the optimal schedulable region, but with backlog compensation as in fair queueing (i.e. if a flow does not have a packet to transmit at any time, it cannot later reclaim this lost slot, since virtual time moves forward). Varying ϱ anywhere between 0 to ∞ allows us the flexibility to control the schedulable region of the scheduler, and provide different levels of delay-bandwidth decoupling.

3.2 Slot Queues and Packet Queues

In most related work on both wireline and wireless fair queueing [1, 3, 10], packets are tagged as soon as they arrive because fair queueing is causal. This works well if we assume no channel error, i.e. a scheduled packet will never be lost. However, in a wireless channel, a lost packet may need to be retransmitted for an error-sensitive flow. Retagging the packet will cause it to join the end of the flow queue and thus cause packets to be delivered out of order.

Fundamentally, there needs to be a separation between 'when to send the next packet', and 'which packet to send next'. The first question should be answered by the scheduler, while the second question is really a flow-specific decision and should be beyond the scope of the scheduler. In order to decouple the answers to these two questions, we use one additional level of abstraction - we tag 'slots' and each slot points to a flow queue. At each time, the scheduler determines which slot will get access to the channel. The HOL packet in the corresponding flow queue is then transmitted. When a packet arrives at a flow queue, a corresponding slot is generated in the slot queue of the flow. The number of slots in the slot queue at any time is exactly the same as the number of packets in the flow queue.

Providing this additional level of abstraction is crucial to being able to achieve the wireless fair service model. Error-sensitive flows will not delete the HOL packet upon channel error during transmission, but delay-sensitive flows may delete the HOL packet once it violates its delay bound. Likewise, the flow may have priorities in its packets, and may choose to discard an already queued packet in favor of an arriving packet when its queue is full [11]. In our scheduling model, we support any queueing and packet dropping policy at the flow level because we decouple slot queues from flow queues.

In following sections, we show how this queueing architecture is also useful in providing compensation.

3.3 Wireless Fair Service in the presence of channel errors - lag, lead and compensation

Thus far, we have assumed an error free channel environment. In this section, we consider how the basic algorithm can be adapted to achieve wireless fair service in a typical wireless network, with location-dependent and bursty channel error and contention.

In the wireless fair service algorithm, we assume fixed size packets (and slots) for convenience.[1] The slots of a flow are tagged as in Section 3.1. In addition to the parameters specified in Section 3.1, each flow i has a 'lead counter' $E(i)$ and a 'lag counter' $G(i)$. The lead counter measures the lead (in slots) of a leading flow while the lag counter measures the lag of a lagging flow. Hence, at least one of these two variables is always 0. The lead of a flow i is bounded by $E_{\max}(i)$, while the lag of i is bounded by $G_{\max}(i)$.

At each time instant, $B(t)$ represents the 'generate set' of the channel, which is the superset of the set of backlogged flows. A flow $i \in B(t)$ iff $Q(i) > 0 \quad OR \quad E(i) > 0$, where $Q(i)$ is the number of packets backlogged in the flow. Thus, slots are generated for a flow either if the flow is backlogged or if the flow is leading.

At each time, the wireless fair service algorithm selects the slot s with the minimum finish tag for transmission, among those slots whose start tag is less than the current virtual time plus the lookahead. The flow i corresponding to this slot may be in one of four states: (1) i is leading, and has designated s for transmission of one of its packets, (2) i is leading, and has designated s for relinquishing to a lagging flow, (3) i is in sync, and (4) i is lagging.

There are a number of questions that need to be answered: (a) how does a leading flow determine when to designate a slot for relinquishing to a lagging flow, (b) which lagging flow gets to transmit in a relinquished slot, and (c) given that the slot and flow selection are made from parts (a) and (b), what is the channel allocation algorithm? In this section, we answer the third question. Questions (a) and (b) are answered in the next section.

As mentioned above, we have four possible states when a slot s corresponding to a flow i has been selected. The following is the channel allocation algorithm in each case:

[1] The generalization of the wireless service algorithm to accommodate variable packet sizes is presented in [6].

1 - Flow i is leading and has designated slot s for transmitting one of its packets: If i perceives a clean channel, then it transmits its HOL packet.

Otherwise, if there is a backlogged lagging flow j which perceives a clean channel, then flow j is allocated the slot, $E(i)$ is decremented by 1, and $G(j)$ is decremented by 1.

Otherwise, if there is a backlogged leading flow j which perceives a clean channel and whose lead is less than its lead bound, i.e. $\exists j, 0 < E(j) < E_{\max}(j)$, then j is allocated the slot, $E(i)$ is decremented by 1, and $E(j)$ is incremented by 1.

Otherwise, if there is any backlogged in sync flow j which perceives a clean channel, then j is allocated the slot, $E(i)$ is decremented by 1, and $E(j)$ is incremented by 1.

Otherwise, if there is any backlogged flow that perceives a clean channel, it is allocated the slot. If there is no backlogged flow with a clean channel, then the slot is wasted.

2 - Flow i is leading and has designated slot s for relinquishing to a lagging flow: If there is a backlogged lagging flow j that perceives a clean channel, then j is allocated the slot, $E(i)$ is decremented by 1, and $G(j)$ is decremented by 1.

Otherwise, if i perceives a clean channel, then it reclaims the slot and transmits its HOL packet.

Otherwise, if there is a backlogged leading flow j which perceives a clean channel and whose lead is less than its lead bound, i.e. $\exists j, 0 < E(j) < E_{\max}(j)$, then j is allocated the slot, $E(i)$ is decremented by 1, and $E(j)$ is incremented by 1.

Otherwise, if there is any backlogged in sync flow j which perceives a clean channel, then j is allocated the slot, $E(i)$ is decremented by 1, and $E(j)$ is incremented by 1.

Otherwise, if there is any backlogged flow that perceives a clean channel, it is allocated the slot. If there is no backlogged flow with a clean channel, then the slot is wasted.

3/4 - Flow i is in sync, or flow i is lagging: If i perceives a clean channel, then it transmits its HOL packet.

Otherwise, if there is a backlogged lagging flow j that perceives a clean channel, then flow j is allocated the slot, $G(i)$ is incremented by 1, and $G(j)$ is decremented by 1.

Otherwise, if there is a backlogged leading flow j which perceives a clean channel and whose lead is less than its lead bound, i.e. $\exists j, 0 < E(j) < E_{\max}(j)$, then j is allocated the slot, $G(i)$ is incremented by 1, and $E(j)$ is incremented by 1.

Otherwise, if there is any backlogged in sync flow j which perceives a clean channel, then j is allocated the slot, $G(i)$ is incremented by 1, and $E(j)$ is incremented by 1.

Otherwise, if there is any backlogged flow that perceives a clean channel, it is allocated the slot. If there is no backlogged flow with a clean channel, then the slot is wasted.

The details for the channel allocation algorithm corresponding to these four states are presented in Figure 1.

Essentially, our slot allocation algorithm tries to avoid wasting slots, tries to avoid disturbing in sync flows, and tries to reduce lag before increasing lead when swapping slots. At all times, the sum of all the lags is equal to the sum of all the leads in the channel.

An interesting situation arises when a lagging flow i clears its packet queue, for example, when all the queued packets violate their delay bounds. In this case, i has no packets that need to be compensated, and we reset $G(i)$ to 0. However, since $\sum_{j \in F} E(j) = \sum_{j \in F} G(j)$, we also need to reduce the lead of all leading flows.[2] We do this in proportion of the lead of the flows,[3] i.e. for each leading flow j, we set $E(j) = E(j).(1 - G(i)/\sum_{f_k \in F} E(k))$.

There are two key properties of our compensation approach: (a) unlike the algorithms in IWFQ and CIF-Q, there is no need to simulate an error-free service in order to determine whether a flow is leading or lagging, and (b) flows that are in sync are unaffected by swapping between leading and lagging flows, and will see the same performance bounds as in their error-free service. Both these properties are very attractive and contribute towards making the wireless fair service algorithm elegantly and efficiently achieve the wireless fair service model.

3.4 Compensation Model

In the previous section, we did not answer two key questions: (a) how does a leading backlogged flow decide whether to designate a scheduled slot for its data transmission or relinquish it for compensation, and (b) which among several lagging backlogged flows gets to transmit in a slot that has been relinquished by a leading flow. We now consider each question in turn.

3.4.1 Rate compensation from leading flows

There are several possible compensation models for leading flows - the simplest one, as adopted by IWFQ, is to relinquish the first $E(i)$ slots, i.e. a leading flow does not transmit any of its data packets till it gets back in sync (or till no lagging flow can transmit a packet in a slot). This compensation model can result in a leading flow being starved out for long periods of time. While bounding the lead is a partial solution, we present a more elegant solution in this paper, that allows for a graceful compensation model.

Consider a leading flow i with a rate weight of r_i, a lead of $E(i)$, and a maximum lead of $E_{\max}(i)$. Flow i hierarchically decomposes itself into two flows, i^c and i^t, with rates of $r_i.E(i)/E_{\max}(i)$, and $r_i.(1 - E(i)/E_{\max}(i))$. i^c is designated to be the compensation flow, while i^t is designated to be the transmission flow. When i is allocated a slot, it hierarchically schedules it among i^c and i^t. All slots belonging to i^c are relinquished.

[2] CIF-Q [10] redistributes the lag of i among *all* backlogged flows. Thus, in-sync flows now become lagging flows, which affects the separation of flows.

[3] An alternative approach is to maintain a virtual lagging flow f_0 that aggregates the weights of all the lagging but non-backlogged flows. When a leading flow i relinquishes a slot in favor of f_0, f_0 simply lets flow i reclaim the slot after decrementing $E(i)$ and $G(0)$ by 1 each. This ensures that $\sum_{j \in F} E(j) = \sum_{j \in F \cup \{f_0\}} G(j)$, and has the same effect as decrementing the lead of all the leading flows without incurring the computational overhead.

```
1  marked_flow = select_flow_with_minimum_tag()
2  if(marked_flow->lead > 0 )
     /* a leading flow : either transmit or give up slot*/
3    if(transmission_slot(marked_flow)) /* transmit */
4      if( slot_state(marked_flow) == CLEAN )
5        return marked_flow;  /* transmit HOL packet */
6      else
7        compensated_flow = select_clean_lagging_flow_from_WRR();
8        if(compensated_flow != NULL )
            /* swap between leading and lagging flows (Case 1)*/
9          marked_flow->lead--;
10         compensated_flow->lag--;
11       else                          /* No clean lagging flow */
12         compensated_flow = pick_leading_flow();
   /* find clean backlogged leading flow with E(i) < E_max(i)*/
13       if( compensated_flow == NULL)
14         compensated_flow = pick_clean_in-sync_flow();
15       if( compensated_flow != NULL)
   /* swap between leading and leading/in-sync flows (Case 1)*/
16     marked_flow->lead --;
17     compensated_flow->lead ++;
18      else
19   compensated_flow = pick_any_clean_backlogged_flow();
20   if (compensated_flow == NULL)
21     wasted_slots ++;
22    return compensated_flow;

23  else  /* compensation_slot => leading flow gives up slot */
24    compensated_flow = select_clean_lagging_flow_from_WRR();
25    if( compensated_flow != NULL )
     /* swap between leading and lagging flows (Case 2) */
26      marked_flow->lead--;
27      compensated_flow->lag--;
28    else                          /* No clean lagging flow  */
29      if( slot_state(marked_flow) == CLEAN )
            /* leading flow reclaims slot for transmission */
30        return marked_flow;
31 else
32       compensated_flow = pick_leading_flow();
   /* find clean backlogged leading flow with E(i) < E_max(i)*/
33       if( compensated_flow == NULL)
34         compensated_flow = pick_clean_in-sync_flow();
35       if( compensated_flow != NULL)
   /* swap between leading and leading/in-sync flows (Case 2)*/
36         marked_flow->lead --;
37         compensated_flow->lead ++;
38     else
39       compensated_flow = pick_any_clean_backlogged_flow();
40       if (compensated_flow == NULL)
41         wasted_slots ++;
42  return compensated_flow;

43 else       /* an in-sync or lagging flow with lag >= 0 */
44   if( slot_state(marked_flow) == CLEAN )
45     return marked_flow;  /* transmit HOL packet */
46   else
47     compensated_flow = select_clean_lagging_flow_from_WRR();
48     if(compensated_flow != NULL )
   /* swap between lagging/in-sync and lagging flows
                                              (Case 3-4) */
49       marked_flow->lag ++;
50       compensated_flow->lag--;
51   else                          /* No clean lagging flow */
52     compensated_flow = pick_leading_flow();
  /* find clean backlogged leading flow with E(i) < E_max(i)*/
53     if( compensated_flow == NULL)
54       compensated_flow = pick_clean_in-sync_flow();
55     if( compensated_flow!=NULL)
   /* swap btw. lagging/in-sync and leading/in-sync flows
 (Case 3-4) */
56       marked_flow->lag ++;
57       compensated_flow->lead ++;
58   else
59     compensated_flow = pick_any_clean_backlogged_flow();
60     if (compensated_flow == NULL)
61       wasted_slots ++;
62 return compensated_flow;
```

Figure 1. Pseudo Code for the Wireless Channel Allocation Algorithm

Note that the decrease in the compensation rate i^c is exponential (see Theorem 4.8 of Section 4.1.5). This has the property of graceful service degradation for a leading flow, as shown in the analysis and simulation in Section 4.

3.4.2 Rate compensation to lagging flows

There are several possible compensation models for lagging flows - the simplest one, as adopted by IWFQ, is to maintain the precedence of slots for lagging flows, and serve the slot with the smallest finish tag among flows that perceive a clean channel. This may lead to the undesirable property that a flow that has been in an error-prone channel for a long time and suddenly perceives a clean channel will capture the channel till it catches up its lag. While bounding the lag is a partial solution to this problem, algorithms such as WPS and CIF-Q have tried to distribute the channel access among lagging flows in a way that prevents a single flow from capturing the channel. In this paper, we present an enhancement to the compensation model proposed in both WPS and CIF-Q.

Our approach is to provide channel access to backlogged lagging flows from two sources: (a) the normal rate-based slot allocation, and (b) distributing the relinquished slots from leading flows fairly among the lagging flows. We maintain a 'compensation weighted round robin' slot allocation among the lagging flows, where the weight of each lagging flow i is $G(i)$. Associated with the compensation WRR is a current pointer, which identifies the next lagging flow in the WRR that should receive a compensation slot. When a leading flow relinquishes a slot, we traverse through the WRR looking for the first flow that perceives a clean channel, and allocate the relinquished slot to that flow. Thus, lagging flow receive slots from two sources at any time t: slots at a rate of $r_i / \sum_{j \in B(t)} r_j$ from the basic wireless fair service algorithm, and slots at a rate of $(G(i)/\sum_{f_k \in F} G(k)).\sum_{f_k \in leading}(E(k)/E_{\max}(k).(r_k/\sum_{j \in B(t)} r_j))$ from the compensation WRR. As is clear from the algorithm, the compensation model allocates the compensation slots fairly among the lagging flows.

3.5 Wireless Medium Access Protocol

In previous sections, we described the various components of the wireless fair service algorithm. In this section, we describe a simple and robust CSMA/CA based wireless medium access protocol that can be used in concert with the wireless fair service algorithm at the base station in order to achieve the wireless fair service model in packet cellular networks.

We describe the MAC protocol in two parts: first we describe the basic access protocol assuming that the base station knows the backlogged uplink and downlink flows, and then we describe how newly generated uplink flows notify the arrival of a new run of packets to the base station.

3.5.1 The basic MAC Protocol Mechanism

The wireless MAC protocol is very simple: we use a CSMA/CA based RTS-CTS-Data-ACK handshake sequence for accomplishing an uplink or downlink data transmission. The only key difference in our protocol is that the mobile host always transmits the RTS packet even if the transmission is downlink. Thus, uplink flows have the following sequence of packets: RTS from the mobile host, CTS from the base station, Data from the mobile host, and ACK from the base

station. Downlink flows have the following sequence of packets: RTS from the mobile host, CTS from the base station, Data from the base station, and ACK from the mobile host.

The base station is assumed to know all the uplink and downlink backlogged flows, and maintains the leads and lags of the flows. Based on the local state at the base station, it can compute the flows that will be allocated the next m slots (where $m = 4$ for implementation; see Section 3.6). Since the base station always sends either the Data or the ACK packet in each data transmission, it piggybacks the identities of the four flows (i.e. the corresponding mobile hosts) in the Data or ACK packet. Stations that have a clean channel will hear the base station.

If a station s_i is identified as the j^{th} transmitter in this sequence ($1 \leq j \leq 4$), it senses the carrier for a time equal to $W_s.j$. If the carrier becomes busy, the station realizes that some station earlier in the transmission sequence has set up communication with the base station. If it senses a free carrier, station s_i sends the RTS packet to the base station. W_s is at least twice the propagation delay in the cell plus the time to send an RTS packet plus some processing time, in order to ensure that each station waits till it can hear the CTS from the base station if some station earlier in the transmission queue initiated the RTS-CTS handshake.

Note that we have assumed that the wireless channel errors are bursty and highly correlated between successive slots. Thus, if a mobile host does not receive the base station's piggybacked transmission list for slot i in the Data or ACK packet of slot $i-1$, then the mobile host assumes that it perceived a channel error and does not contend for the next slot. Hence, only stations which are in the transmission list for slot i and perceive a clean channel in slot $i-1$ will contend for sending the RTS in slot i.

In our architecture, the mobile host always initiates the RTS handshake because the base station does not know which mobile hosts perceive a clean channel. Having the mobile hosts initiate the RTS eliminates the need for the base station to poll mobile hosts to see if they have a clean channel.

Thus, we can support the wireless fair service algorithm efficiently and in a robust manner, with a simple modification to the basic CSMA/CA wireless medium access algorithm.

3.5.2 Notifying the base station about backlogged flows

A flow may change its backlog status under three conditions: (a) it was previously not backlogged but received a new run of packets, (b) it just finished transmission of the last packet in a run, and (c) it deleted all its backlogged packets because they violated their delay bounds.

Case (b) is simple; when an uplink flow transmits a packet, it has a flag to notify the base station if the flow has more packets backlogged or not. We handle case (c) as follows: when the base station identifies the flow for transmission in the next slot, the corresponding mobile host sends an NRTS packet instead of an RTS packet. Upon receiving an NRTS packet, the base station does not send a CTS, and deletes the flow from its backlogged list. Case (a) is the hard case, because when a flow gets backlogged, the mobile host should notify the base station immediately. When this situation happens, the mobile host waits for the completion of the current packet transmission, and aggressively sends an RTS without waiting for any time with a probability of p (note that the first station identified in the transmission list in the previous slot will sense the carrier for W_s time before initiating the RTS from Section 3.5.1). Thus, notification of a new run of packets is p-persistent and occurs immediately after the completion of the current packet. Typically, we expect that the number of stations that generate a new run of packets during each packet transmission is very small, thus the probability of collision during the new run notification is very small. When collisions do happen, the persistence probability p performs binary exponential backoff till all new run notifications are successfully transmitted to the base station. Once the base station has the information about the newly backlogged flows, it performs the wireless fair service algorithm as detailed in the previous section.

3.6 Implementation Complexity

It should be noted that if the designated flow is able to transmit at each time instant, i.e., the selected flow perceives a clean channel at the time that it is being selected, then our algorithm has the same computational complexity as the WFQ algorithm [1]. In fact, many computationally efficient approximations to WFQ, such as STFQ [3] and SCFQ [7], can be directly applied to our framework.

However, if the selected flow cannot transmit in the designated slot due to channel error, then the algorithm may need to search through a pre-specified number of flows (m) in order to choose the first flow with a clean channel for transmission. In the extreme case, $m = N$, where N is the number of flows. The choice of m is a tradeoff between computational complexity and the channel slot utilization (i.e. how many wasted slots can be tolerated).

4 Analysis and Simulation of the Wireless Fair Service Algorithm

4.1 Analysis

In this section, we analyze the wireless fair service algorithm described in Section 3 assuming perfect MAC and clean channels.[4] By clean channels, we mean that over a time window $[t_1, t_2]$, all flows will perceive error-free channel states. Note that this scenario might happen after some flows experience error-prone channels before t_1. A detailed analysis of the algorithm on error-prone channels over $[t_1, t_2]$ (assuming the generic channel model described in Section 2.1) is given in [6]. We omit it here due to lack of space. We further impose the following two restrictions:

- The lookahead parameter is set as $\varrho = \infty$.
- r_i is the normalized rate weight for flow i, i.e. $\sum_{i \in F} r_i = 1$ where F denotes the set of total flows scheduled at this server. Similarly, we also normalize Φ_i such that $\sum_{i \in F} \Phi_i = 1$.

We denote the credit of flow i at t as $C_i(t)$ (in bits), and the maximum credit bound (in bits) for flow i as $C_i^{\max}$. According to Section 3.3, we have $C_i(t) := E(i)L_p > 0$ for leading flows, $C_i(t) := -G(i)L_p < 0$ for lagging flows, and $C_i(t) = 0$ for in sync flows, where L_p is the slot size (in bits).

4.1.1 Schedulability Condition for Error-Free Service Model

We adopt the same notations as in [2]. Let the vector $\vec{D} = (\bar{D}_1, \bar{D}_1, \ldots, \bar{D}_N)$ be a list of required upper bounds on delay so that no packet from flow i is delayed by $\bar{D}_i$. The vector $\vec{D}$ is *schedulable* under a scheduling policy π if for all arrival

[4] For proofs of the theorems presented, please refer [6].

patterns that satisfy the stability condition, and for each flow $i \in F$, no packet of flow i is delayed by more than D_i (measured in seconds or bits). The *schedulable region* of a policy π is the set of all vectors schedulable under π. A delay-optimal policy has the *largest* schedulable region among a class of admissible scheduling policies.

Theorem 4.1 *Our algorithm is delay-optimal among the class of non-preemptive policies in virtual time domain when $\varrho = \infty$ and the channels are error free at all times. The schedulable region of the algorithm consists of the set of vectors which satisfy the constraints*

$$(k+1)L_{\max} \le \bar{D}_k(1 - \sum_{n=1}^{k-1} r_n) + \sum_{n=1}^{k-1} r_n \bar{D}_n, \ 1 \le k \le N-1 \quad (1)$$

$$NL_{\max} \le \bar{D}_N(1 - \sum_{n=1}^{N-1} r_i) + \sum_{n=1}^{N-1} r_n \bar{D}_n \quad (2)$$

whenever $L_{\max} \le \bar{D}_1 \le \bar{D}_2 \le \ldots \le \bar{D}_N$, where $\bar{D}_i = \frac{L_{\max}}{\Phi_i}$ with $L_{\max}$ denoting the maximum packet size (in bits). □

4.1.2 Throughput Guarantees

Theorem 4.2 *Let a flow i be continually backlogged over a real time interval $[t_1, t_2]$. Assume that all flows have clean channels over time interval $[t_1, t_2]$. Then, the aggregate service $W_i(t_1, t_2)$ (in bits) that flow i receives during time interval $[t_1, t_2]$, is bounded by*

$$W_i(t_1, t_2) \ge r_i C(t_2 - t_1) - r_i \sum_{i \in F} L_{\max} - L_{\max} - \alpha L_{\max} \frac{r_i}{\Phi_i} + \beta_i \quad (3)$$

where $\alpha = 2$ if flow i is in sync and no packet of flow i was served at t_1^-; $\alpha = 1$, otherwise. $\beta_i = 0$ for in sync flows and

$$\beta_i = -r_i C \int_{t_1}^{t_2} \frac{C_i(t)}{C_i^{\max}} dt \qquad \text{if flow } i \text{ is leading;}$$

$$\beta_i = \min\left\{ |C_i(t_1)|, \sum_{k \in \mathcal{L}(t_1)} \frac{|C_k(t_1)| r_k C(t_2 - t_1)}{r_k C(t_2 - t_1) + |C_i^{\max}|} \frac{|C_i(t_1)|}{\sum_{j \in \mathcal{G}(t_1)} |C_j(t_1)|} \right\},$$

if flow i is lagging.

in which $\mathcal{L}(t)$ denotes the set of leading flows at t, and $\mathcal{G}(t)$ denotes the set of lagging flows at t. □

Remark 4.1 *The theorem shows that we do achieve the decoupling of delay and throughput in the sense that the long-term throughput is mainly determined by the rate weight parameter r_i, not the delay weight parameter Φ_i.* □

4.1.3 Delay Guarantees

For an arbitrary packet of flow i, usually the delay guarantee is defined with respect to the *expected arrival time* [3]. The expected arrival time $EAT(p_i^{k+1})$ of $(k+1)^{th}$ packet from flow i is formally defined as

$$EAT(p_i^{k+1}) = \max\left\{ A(p_i^{k+1}), \quad EAT(p_i^k) + \frac{L_i^k}{r_i C} \right\}, \quad k \ge 0.$$

where $A(p_i^{k+1})$ and L_i^k are the arrival time and packet length of $(k+1)^{th}$ packet from flow i, respectively.

Theorem 4.3 *(Leading flows) As long as a flow i is leading over $[t_1, t_2]$, its maximum delay with respect to its expected arrival time is zero.* □

Theorem 4.4 *(In-sync flows) The new queue delay d_i^{nq} for an in-sync flow i, defined as the difference between the departure time (PDT) and the expected arrival time(EAT) of its head-of-line (HOL) packet, is given by*

$$d_i^{nq} \le \frac{L_{\max}}{C\Phi_i} + (\sum_{j \in F} L_{\max})/C \quad (4)$$

where F denotes the set of all flows. □

Remark 4.2 *As we can see from (4), the packet delay bound is determined by the delay weight parameter Φ_i, not the rate weight r_i. This also illustrates how delay is decoupled from the throughput quantitatively.* □

Theorem 4.5 *(Lagging flows) Consider a backlogged lagging flow i at t, which has a credit $C_i(t)$ (in bits). Assume that all flows have a uniform packet size L_p, and all have clean channels and are backlogged from time t, and $\sum_{j \in F} r_j = 1$ without any loss of generality, where F denotes the total set of flows. Then the HOL packet delay of flow i is upper bounded by*

$$d_i^{nq} \le \left(\sum_{j \in F} L_p / C \right) + \frac{L_p}{C\Phi_i} + \frac{|C_i(t)|}{Cr_i} + \min\left\{ \frac{L_p}{r_i C}, T_m \right\} \quad (5)$$

where T_m is calculated by

$$CT_m \sum_{k \in \mathcal{L}(t)} \frac{|C_k(t)| r_k / L_p}{r_k C T_m + |C_i^{\max}|} = \frac{\sum_{j \in \mathcal{G}(t)} |C_j(t)|}{|C_i(t)|} \quad (6)$$

in which $\mathcal{L}(t)$ denotes the set of leading flows at t, and $\mathcal{G}(t)$ denotes the set of lagging flows at t. □

4.1.4 Fairness Guarantees

Theorem 4.6 *(Long-term Fairness Index) For a continually backlogged flow i over clean channels, it achieves the following long-term throughput fairness index:*

$$\lim_{v \to \infty} \frac{W_i(0, v)}{v} = r_i \quad (7)$$

Theorem 4.7 *(Short-term Fairness) For any interval $[v_1, v_2]$ where v_1 and v_2 are virtual times measured in bits, in which two in-sync flows i and j perceive clean channels and are continually backlogged on $[v_1, v_2]$, the difference in the service received by two flows is given by*

$$\left| \frac{W_i(v_1, v_2)}{r_i} - \frac{W_j(v_1, v_2)}{r_j} \right| \le \frac{L_{\max}}{r_i} + \frac{L_{\max}}{r_j} + \frac{\alpha L_{\max}}{\min(\Phi_i, \Phi_j)} \quad (8)$$

where $\alpha = 1$ if i and j have been receiving services at v_1^-; and $\alpha = 2$ if no packet of flow i or j was served at v_1^-. □

4.1.5 Graceful Service Degradation

Theorem 4.8 *(Graceful service degradation for a leading flow) Consider a leading backlogged flow i over a time interval $[t_1, t_2]$. Assume $C_i(t_1) = C_0$ and there always exists a lagging flow which can take the compensation slot whenever flow i gives up a compensation slot throughout $[t_1, t_2]$. Then, for any time $t \in [t_1, t_2]$, its credit $C_i(t)$ is given by*

$$C_i(t) = C_0 e^{-\frac{r_i C}{C_i^{\max}}(t-t_1)} , \quad (9)$$

its instantaneous transmission rate $r_i(t)$ is given by

$$r_i(t) = r_i \left(1 - \frac{C_0}{C_i^{\max}} e^{-\frac{r_i C}{C_i^{\max}}(t-t_1)}\right) , \quad (10)$$

and the maximum graceful degradation time t_g is given by

$$t_g = t_1 + \frac{C_i^{\max}}{r_i C} \ln C_0 \quad (11)$$

Remark 4.3 *As we can see from (9) and (10), both the lead (given by $C_i(t)$) and the instantaneous compensation rate (given by $r_i - r_i(t)$) of a leading flow decreases exponentially. This is different from the degradation schemes in WPS [8] and CIF-Q [10].*

4.2 Simulation Results

This section presents the simulation results for the wireless fair service algorithm. As described in the previous sections, the key features of our algorithm are the following: separation between flows, decoupling of rate and delay, optimal schedulable region, short term throughput and fairness guarantees for error-free flows, long term throughput and fairness guarantees for all flows, and graceful service degradation for leading flows. In order to illustrate the effect of each of these features, we present some examples in this section.

The following performance measures are used to evaluate the algorithm. W: number of transmitted packets of the flow expressed as a fraction of the total number of packets transmitted for all flows; P_l: loss probability, i.e. fraction of packets dropped; $D_{\max}$: maximum delay of successfully transmitted packets; D_{avg}: average delay of successfully transmitted packets; σ_D: standard deviation of the delay; d^{nq}: maximum new queue delay. Note that the delay and throughput parameters are expressed in terms of slots.

Each of our simulations had a typical run of 50000 time units. We averaged each result over 25 simulation runs. To obtain measurements over short time windows, we measured the parameters over 5 different time windows, of size 200 time units each, in a single simulation run, and averaged the values got over 5 distinct simulation runs, for the results shown here.

We have considered Poisson sources and MMPP sources in our simulations. For the MMPP sources, the modulated process is a continuous-time Markov chain which is in one of two states ON or OFF. The transition rate from the ON to OFF is 0.9 and OFF to ON is 0.1.

The wireless channel in our simulations evolves according to a two-state discrete Markov chain. p_g is the probability that the next time slot is good given that the current time slot is in error, and p_e is the probability that the next time slot is in error given that the current slot is good.

We use one-step prediction for the channel state, i.e., the channel state for the current time slot is predicted to be the same as the monitored channel state during the previous time slot. Though this is obviously not perfect, our simulation results show that it is reasonably effective for typical wireless channel error models.

	Src	r_i	Φ_i	W	P_l	$D_{\max}$	D_{avg}	σ_D	d^{nq}
I	1	0.11	0.11	0.11	0	76.5	8.7	10	13
	2	0.44	0.44	0.44	0	40.1	3.8	4.8	2.1
	3	0.44	0.44	0.44	0	40.2	3.8	4.8	2.1
II	1	0.11	0.11	0.11	0	22.5	8.4	6.8	6.7
	2	0.44	0.44	0.44	0	10.7	3.1	1.7	1.4
	3	0.44	0.44	0.44	0	11.1	3.1	1.4	1.7

Table 1: Source parameters and results for Example 1: Scenario (a).

We present four examples in this section, where each one is used to demonstrate a specific feature of our algorithm. Example 1 illustrates the decoupling of rate and delay, thus expanding the schedulable region. Example 2 shows the performance of error-sensitive and delay-sensitive flows in an error-prone channel. Example 3 demonstrates the graceful degradation of leading flows during compensation. It also illustrates how in-sync flows are not disturbed in the presence of leading flows. Example 4 shows how an adaptive source can maintain its throughput, even when it drops packets due to channel error or delay violation.

Example 1: Decoupling of rate and delay. Here, we illustrate that our algorithm has a larger schedulable region than IWFQ [8] and CIF-Q [10], due to the decoupling of delay and bandwidth. Consider three Poisson sources with error-free channels. Source 1 has an average rate of 0.11, Sources 2 and 3 have average rates of 0.44 each. We consider two scenarios:

(a) In this scenario, we set the delay weights Φ_i to be equal to the rate weights r_i of the sources, and the lookahead $\varrho = \infty$. Our algorithm is now the same as the Wireless Fair Queueing algorithm [8]. The parameters and the simulation results over the entire run (I) and over small time windows (II) are given in Table 1. As expected, the rates obtained by the sources are proportional to their weights, and the configuration is schedulable. [5]

(b) Now, we change the delay weights for each of the sources, setting $\Phi_1 = 0.9$, $\Phi_2 = 0.09$ and $\Phi_3 = 0.009$. The simulation results over the entire run (I), and over small time windows (II) are shown in Table 2. We can see that Source 1, which has a larger delay weight than the other sources, experiences a much smaller delay, even though its rate is smaller than the other two sources. On the other hand, Source 3 has a large rate, but it sees a large delay, as it has a smaller delay weight. Thus, the Wireless Fair Service algorithm can schedule low rate, low delay flows, as well as high rate, high delay flows, due to delay-bandwidth decoupling. As shown in Theorem 4.1, our algorithm has the optimal schedulable region.

Example 2: Error-sensitive vs. Delay-sensitive flows. We now consider an example to show how our algorithm performs when the channel is error-prone, and when the flows can be delay-sensitive or error-sensitive. A delay-sensitive flow drops its packets when the packets are in the queue for a time larger than the specified delay bound. An error-sensitive flow drops packets when it tries to transmit a packet

[5] Note that the delays are not inversely proportional to the Φ_i values because of a number of reasons: not all flows are backlogged at any time; the delay is dependent on the queue size, etc.

	Src	r_i	Φ_i	W	P_l	$D_{\max}$	D_{avg}	σ_D	d^{nq}
I	1	0.11	0.9	0.11	0	37.5	1.0	2.7	15.8
	2	0.44	0.09	0.44	0	40.8	2.9	4.4	22.8
	3	0.44	0.009	0.44	0	64.7	6.8	7.4	29.9
II	1	0.11	0.9	0.11	0	5.4	0.8	1.4	4
	2	0.44	0.09	0.44	0	11.8	2.6	2.9	5
	3	0.44	0.009	0.44	0	20.1	6.6	5.1	7

Table 2: Source parameters and results for Example 1: Scenario (b).

	Src	W	P_l	$D_{\max}$	D_{avg}	σ_D	d^{nq}
Entire Run	1	0.327	0	165.0	19.33	22.45	91.5
	2	0.325	0	147.5	19.76	23.28	94.3
	3	0.348	0	19.5	1.76	2.38	19.5
Small Window	1	0.328	0	26.0	8.38	7.41	15.0
	2	0.323	0	26.7	8.92	10.32	17.5
	3	0.351	0	7.8	1.39	1.93	7.71
Error free channels	1	0.328	0	46.3	5.15	5.31	3.11
	2	0.327	0	46.1	5.12	5.17	3.11
	3	0.345	0	2.89	1.59	1.7	2.89

Table 3: Results for Example 2: Error-sensitive flows.

for a maximum number of times and encounters a channel error on all its attempts.

We consider three sources, where Sources 1 and 2 are Markov-modulated Poisson processes (MMPPs) and the arrivals occur according to a Poisson process of rate 1.5 when the Markov chain is in the ON state, and Source 3 is a constant source with a rate of 0.25 (i.e packet inter-arrival time of 4). The channel for Sources 1 and 2 evolve according to a two-state discrete Markov chain having a steady state error probability $P_E = 0.3$ with $p_g + p_e = 0.1$. Source 3 has an error-free channel. The rate weights for all sources are $r_i = 0.333$. The delay weights are also assigned to be equal to the rate weights. We consider two cases:

(a) Error-sensitive flows: For each packet, we limit the maximum number of retransmissions to 8, i.e. a packet is dropped if it is not successfully transmitted after nine attempts. The simulation is performed over an entire run, as well as over a set of small time windows.

The simulation results are presented in Table 3. For purposes of comparison, we also present the simulation results for same set of flows with error-free channels. It illustrates the fact that we get the same rates as in an ideal error-free channel with fair queueing, but since the channel is error-prone for some flows, those flows have larger delays. All flows get the rates in proportion to their rate weights. From the results with the small time windows, we see that our compensation model provides for short term fairness and throughput guarantees even when the channels are error-prone.

(b) Delay-sensitive flows: Instead of setting the maximum number of retransmission attempts per packet, we set an upper limit on the maximum delay of a packet to be 100. If a packet is in the system for more than 100 time slots, then it is dropped; this could possibly happen even before it reaches the head-of-the-queue. Thus, our sources are now delay-sensitive, rather than error-sensitive.

We present the simulation results in Table 4. We also present the performance metrics over short time windows. This example complements the results of case (a) by leading to the same conclusions regarding the short term fairness and rate guarantees. It is also seen that Source 3, which has an error-free channel, does not experience a change in its throughput, due to the compensation given to other flows. Thus, we are able to achieve separation of flows.

	Src	W	P_l	$D_{\max}$	D_{avg}	σ_D	d^{nq}
Entire run	1	0.326	0.005	99.9	17.0	19.3	88.0
	2	0.326	0.006	100.0	17.6	19.8	88.2
	3	0.348	0	19.3	2.0	2.7	6.4
Small time window	1	0.328	0	33.56	8.7	9.2	22.3
	2	0.325	0	25.78	9.5	7.6	21.9
	3	0.347	0	6.0	1.1	1.6	5.9

Table 4: Results for Example 2: Delay-sensitive flows.

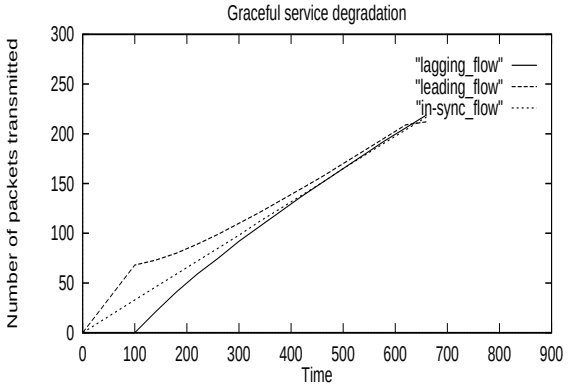

Figure 2: Graceful Service Degradation

Example 3: Graceful service degradation. In this example, we demonstrate the graceful degradation of leading flows. There are three flows all with identical delay and rate weights: Flow 1 is in error till time $t = 100$. Flows 2 and 3 are always error-free. All flows are backlogged at any instant of time. We bound the $E_{\max}$ and $G_{\max}$ of each flow to be 50. Figure 2 presents the plot of the number of packets served over time.

We are able to see that up to time t=100, Flow 1 is starved as it encounters channel error. Flow 2 receives the excess service. Due to the inherent nature of our algorithm that avoids disturbing in-sync flows as much as possible, we see that Flow 3 does not receive any of the excess service. At time $t = 100$, Flow 1 has accumulated a lag of 50, Flow 2 has a lead of 50, while Flow 3 does not have any lead or lag. After t=100, lagging Flow 1 experiences a clean channel and it starts reducing its lag. Leading Flow 2 gives up some of its slots to Flow 1, and we can see that the compensation decreases exponentially. Thus, even though Flow 2 had accumulated a lot of lead, it does not get starved, and observes a graceful degradation of service, while in-sync Flow 3 is not disturbed at all. Flow 1 also sees a graceful increase of rate to 0.33 as it reduces its lag to 0.

In Idealized Wireless Fair Queueing [8], from time $t = 100$, the curve would have levelled off for the leading flow and the in-sync flow, since the lagging flow would have had a higher precedence with its lower tags. This would happen till the lagging flow caught up with the leading and in sync flows.

CIF-Q [10] uses the parameter α to control how soon a leading flow gives up its lead, but the service degradation achieved is linear. In our algorithm, we use the parameter $C_{\max}(i)$ to determine how soon a leading flow gives up its lead, and we achieve exponential degradation of service.

Example 4: Adaptive Sources A delay-sensitive flow that drops its packets when they exceed their delay bounds (due to channel error) will cease to be backlogged and thus lose its

	p_g	p_e	P_E
Flow 2	0.07	0.03	0.3
Flow 3	0.05	0.05	0.5

Table 5: Channel parameters for Example 4

	$D_{\max}$	Non-adaptive	Adaptation window				
			100	50	40	30	10
W	∞	.3324					
	100	.3275	.3283	.3291	.3304	.3308	.3319
	50	.3082	.3265	.3268	.3273	.3292	.3298

Table 6: Effect of adaptive nature of source on throughput

compensation. A flow can react to this packet dropping by generating additional packets equal to the number of packets lost at a higher rate.

In this example, we look at the effect of the latency of adaptation on the throughput for a flow in the presence of channel error. In our simulation, we have incorporated a time-window for a flow that determines how soon a flow reacts to this packet dropping. A window of 10 implies that when a source generates excess packets in reaction to a packet dropping, it will be 10 time units after the drop is observed. Ideally, this time-window should be 0. In this example, we measure the impact of the latency of adaptation on throughput. In particular, we have tried to show that the faster a flow adapts to packet dropping due to delay violations, the lesser is the decrease in throughput observed. Let us consider three flows: Flow 1 has an error-free channel at all times, Flow 2 and Flow 3 perceive channel errors according to Table 5. All flows are MMPP sources with the Poisson arrival rate as 1.2 when the Markov chain is in the ON state. All three flows are delay-sensitive with the delay bound to be 100.

Table 6 shows the throughput obtained for Flow 3 as a fraction of the overall throughput for different values of this time-window. The results show that the throughput increases as time-windows become smaller, i.e., when Flow 3 becomes more adaptive with respect to the rate. We see a two percent increase in throughput compared to the case when Flow 3 is non-adaptive, when the delay bound is set to be 100. If we reduce the delay bounds further (implying a greater number of losses), we see up to seven percent increase in throughput.

5 Conclusions

Emerging indoor and outdoor packet cellular networks will seek to support diverse communication-intensive applications with sustained quality of service requirements over scarce, dynamic and shared wireless channels. While fair queueing has long been a popular paradigm for supporting bounded delay access (and hence guaranteed QoS) to a shared wireline link, wireline fair queueing algorithms cannot be applied directly to the wireless domain because they do not address issues of location-dependent and bursty channel error, channel contention, dynamic channel capacity, shared broadcast channels for uplink and downlink flows, etc.

In this paper, we present a wireless fair service algorithm which provides short-term fairness among flows which perceive a clean channel, worst-case delay bounds for packets, short-term throughput bounds for flows with clean channels and long-term throughput bounds for all flows with bounded channel error, optimal schedulable region by decoupling delay/bandwidth weights, and supports both delay sensitive and error sensitive data flows. The practical implementation of this algorithm in a packet cellular environment is accomplished within the framework of the simple and robust CSMA/CA medium access protocol. We provide the analytical properties of the the wireless fair service algorithm and illustrate them through simulations.

Ongoing work seeks to refine the analytical bounds of this algorithm, perform more extensive simulations and instantiate it in an implementation testbed.

References

[1] A. Parekh, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks," *PhD Thesis*, MIT Laboratory for Information and Decision Systems, Technical Report LIDS-TR-2089, 1992.

[2] L. Georgiadis, R. Guerin and A. Parekh, "Optimal multiplexing on a single link: delay and buffer requirements," *IEEE Trans. Information Theory*, 43(5), pp. 1518-1535, September 1997.

[3] P. Goyal, H. Vin and H. Cheng, "Start-time fair queuing: a scheduling algorithm for integrated services packet switching networks," *ACM SIGCOMM'96*, August 1996.

[4] J.C.R. Bennett and H. Zhang, "WF^{2}Q: Worst-case fair weighted fair queueing," *IEEE INFOCOM'96*, March 1996.

[5] P. Bhagwat, P. Bhattacharya, A. Krishma and S. Tripathi, "Enhancing throughput over wireless LANs using channel state dependent packet scheduling," *IEEE INFOCOM'97*, April 1997.

[6] V. Bharghavan, S. Lu and N. Thyagarajan, "Wireless Fair Service Algorithm: Algorithm and Properties," Technical Report, UIUC, *http://timely.crhc.uiuc.edu*, April 1998.

[7] S. Golestani, "A self-clocked fair queueing scheme for broadband applications," *IEEE INFOCOM'94*, June 1994.

[8] S. Lu, V. Bharghavan and R. Srikant, "Fair scheduling in wireless packet networks," *ACM SIGCOMM'97*, August 1997.

[9] C. Chang, J. Chang, K. Chen and M. You, "Guaranteed quality-of-service wireless access to ATM," *IEEE Journal on Selected Areas in Communications*, 15(1), January 1997.

[10] T.S. Eugene Ng, I. Stoica and H. Zhang, "Packet fair queueing algorithms for wireless networks with location-dependent errors," *IEEE INFOCOM'98*, March 1998.

[11] S. Ha, K.W. Lee and V. Bharghavan, "Performance evaluation of scheduling algorithms in integrated services packet networks," *the Third IEEE Symposium on Computers and Communications*, July 1998.

[12] P. Ramanathan and P. Agrawal, "Adapting Packet Fair Queueing Algorithms to Wireless Networks," *ACM MOBICOM'98*, October 1998.