Fair Queueing in Wireless Networks: Issues and Approaches

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

Abstract

Fair queueing has long been a popular paradigm for providing bounded delay channel access and separation between flows in wireline networks. 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 channel error.

In this article, we identify the key issues in wireless fair queueing, define a wireless fair service model, present a generic framework for designing wireless fair queueing algorithms, and explore the solution space within this framework.

Using simple examples, we show that some of the wireless fair queueing algorithms currently proposed in literature can achieve wireless fair service.

1 Introduction

Recent years have witnessed a tremendous growth in the wireless networking industry. With the increasing usage of wireless networks in both indoor and outdoor computing environments, the issue of providing fair channel access among multiple contending hosts or packet flows over a scarce and shared wireless channel has come to the fore. In wireline networks, *fluid fair queueing* has long been a popular paradigm for providing bounded delay channel access and fairness among packet flows over a shared uni-directional link [1]. However, adapting fair queueing to the wireless domain is non-trivial because of the unique problems in wireless channels. These problems include location-dependent and bursty errors, channel contention, and joint scheduling of uplink and downlink flows. Consequently, the fair queueing algorithms proposed in literature for wireline networks do not apply directly to wireless networks.

Recently, a number of algorithms have been proposed for adapting fair queueing to the wireless domain. In fluid fair queueing, during each infinitessimally small time window, the channel bandwidth is distributed fairly among all the backlogged flows, i.e. among the flows that have data to transmit during that time window. However, in the wireless domain, a packet flow may experience location-dependent channel error and hence may not be able to transmit or receive data during a given time window. The goal of wireless fair queueing algorithms is to make short bursts of location-dependent channel error transparent to users by a dynamic reassignment of channel allocation over small timescales. Specifically, a backlogged flow f that perceives channel error during a time window $[t_1, t_2]$ is compensated over a later time window $[t_1^{'}, t_2^{'}]$ when f perceives a clean channel. Compensation for f involves granting additional channel access to f during $[t_1^{'}, t_2^{'}]$ in order to make up for the lost channel access during $[t_1, t_2]$, and this additional channel access is granted to f at the expense of flows that were granted additional channel access during $[t_1, t_2]$ while f was unable to transmit any data. Essentially, the idea is to swap channel access between a backlogged flow that perceives channel error and backlogged flows that do not, with the intention of reclaiming the channel access for the former when it perceives a clean channel. The different proposals differ in terms of how the swapping takes place, between which flows the swapping takes place, and how the compensation model is structured.

While fair queueing is certainly not the only paradigm for achieving fair and bounded delay access in shared channels, this article focuses exclusively on the models, policies, and algorithms for wireless fair queueing. In Section 2, we describe the network and wireless channel model, the wireless service model, fluid fair queueing, and then outline the major issues in channel-dependent scheduling. In Section 3, we discuss different policies and mechanisms for swapping, compensation, and achieving short-term and long-term fairness in wireless fair queueing. In Section 4, we provide an overview of several contemporary algorithms for wireless fair queueing. In Section 5, we illustrate the properties of wireless fair queueing with some simple examples. Section 6 concludes this article.

2 Models and Issues

In this section, we first describe the network and channel model, and the service model for wireless fair queueing considered in this article. We then provide a brief overview of wireline fluid fair queueing, and outline the key issues that need to be addressed in order to adapt fluid fair queueing to the wireless domain.

2.1 Network and Channel Model

The technical discussions presented in this article are specific to a packet cellular network consisting of a wired backbone and partially overlapping wireless cells. Each cell is served by a base station that performs the scheduling of packet transmissions for the cell (see Figure 1). Neighboring cells are assumed to transmit on different logical channels. All transmissions are either uplink (from a mobile host to a base station) or downlink (from a base station to a mobile host). Every mobile host in a cell can communicate with the base station, though it is not required for any two mobile hosts to be within range of each other. Each flow of packets is identified by a <host, uplink/downlink flag, flow id> triple.

The distinguishing characteristics of the network model under consideration are the following:

- the *channel capacity is dynamically varying*
- *channel errors are location-dependent and bursty* in nature

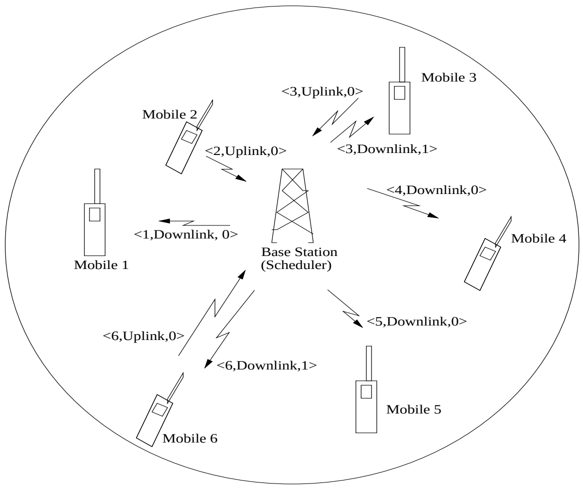

Figure 1: Cellular Architecture

- there is *contention* for the channel among multiple mobile hosts
- *mobile hosts do not have global channel state* (in terms of which other hosts contending for the same channel have packets to transmit, etc.)
- the scheduling must take care of both *uplink and downlink flows*
- 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.

In terms of the wireless channel model, we consider a single channel for both uplink and downlink flows, and for both data and signaling. We assume that each packet transmission involves a RTS-CTS handshake between the mobile host and the base station that precedes the data transmission. Successful receipt of a data packet is followed by an acknowledgement. At most one packet transmission can be in progress at any time in a cell (following the popular CSMA/CA paradigm for wireless medium access). Even though all the mobiles and the base station share the same channel, stations may perceive different interference and fading patterns. Therefore, errors are location dependent. For simplicity of the discussion, we consider packet transmissions to be of fixed size that can be transmitted in one *slot*, though the algorithms described in this article are applicable for variable size packet transmissions as well.

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 typically 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 data or an acknowledgement. Thus, every host that perceives a clean channel must be able to overhear some packet from the base station during each transmission. Note that while we use the CSMA/CA paradigm as a specific instance of a wireless medium access protocol, this is not a requirement in terms of the applicability of the wireless fair queueing algorithms described in this article. In fact, the issues that need to be addressed in medium access are somewhat orthogonal to the issues that need to be addressed in wireless fair queueing [6, 7].

2.2 Service Model

Wireless fair queueing seeks to provide the same service to flows in a wireless environment as traditional fair queueing does in wireline environments. This implies providing bounded delay access to each flow, and providing full separation between flows. Specifically, fluid fair queueing can provide both long-term fairness and instantaneous fairness among backlogged flows. However, we show in Section 2.4 that in the presence of location-dependent channel error, the ability to provide both instantaneous and long-term fairness will be violated. Channel utilization can be significantly improved by swapping channel access between error-prone and error-free flows at any time - this will provide long term fairness, but not instantaneous fairness even in the fluid model in wireless environments. Since we need to compromise on complete separation[1] between flows in order to improve efficiency, wireless fair queueing necessarily provides a somewhat less stringent quality of service than wireline fair queueing.

We now define the *wireless fair service model* that wireless fair queueing algorithms typically seek to satisfy, and defer the discussion of the different aspects of the service model to subsequent sections. The wireless fair service model has the following properties:

- *short-term fairness* among flows that perceive a clean channel and *long-term fairness* for flows with bounded channel error
- *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
- support for both *delay sensitive and error sensitive* data flows

We define the *error-free service* of a flow as the service that it would have received at the same time instant if all channels had been error-free, under identical offered load. A flow is said to be *leading* if it has received channel allocation in excess of its error-free service. A flow is said to be *lagging* if it has received channel allocation less than its error-free service. If a flow is neither leading nor lagging, it is said to be *in sync*, since its channel allocation is exactly the same as its error-free service. If the wireless scheduling algorithm explicitly simulates the error-free service, then the lead and lag can be easily computed by computing the difference of the queue size of a flow in the error-free service and the queue size of the flow in reality. If the queue size of a flow in the error-free service is larger, then the flow is leading. If the queue size of a flow in the error-free service is smaller, then the flow is lagging. If the two queue sizes are the same, then the flow is *in sync*.

[1]Separation denotes the degree to which the service of one flow is unaffected by the behavior and channel conditions of another flow.

2.3 Fluid Fair Queueing

As background reference, we now provide a brief overview of fluid fair queueing in wireline networks.

Consider a uni-directional link that is being shared by a set F of data flows. Consider also that each flow $f \in F$ has a *rate weight* r_f. At each time instant t, the rate allocated to a backlogged flow f is $r_f C(t)/\sum_{i \in B(t)} r_i$, where $B(t)$ is the set of non-empty queues and $C(t)$ is the link capacity at time t. Therefore, fluid fair queueing serves backlogged flows in proportion to their rate weights. Specifically, for any time interval $[t_1, t_2]$ during which there is no change in the set of backlogged flows $B(t_1, t_2)$, the channel capacity granted to each flow i, $W_i(t_1, t_2)$, satisfies the following property:

$$\forall i, j \in B(t_1, t_2), \quad \left| \frac{W_i(t_1, t_2)}{r_i} - \frac{W_j(t_1, t_2)}{r_j} \right| = 0. \tag{1}$$

The above definition of fair queueing is applicable both for channels with constant capacity and for channels with time varying capacity.

Since packet switched networks allocate channel access at the granularity of packets rather than bits, packetized fair queueing algorithms must approximate the fluid model. 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 $[t_1, t_2]$. For example, Weighted Fair Queueing (WFQ) [2] and Packet Generalized Processor Sharing (PGPS) [1] are non-preemptive packet fair queueing algorithms that simulate fluid fair queueing and transmit that packet whose last bit would be transmitted earliest according to the fluid fair queueing simulation.

In WFQ, each packet is associated with a start tag and finish tag, which correspond respectively to the 'virtual time' at which the first bit of the packet and the last bit of the packet are served in fluid fair queueing. The scheduler then serves the packet with the minimum finish tag in the system. The k^{th} packet of flow i that arrives at time $A(p_i^k)$ is allocated a start tag, $S(p_i^k)$, and a finish tag, $F(p_i^k)$ as follows:

- $S(p_i^k) = max\{V(A(p_i^k)), F(p_i^{k-1})\}$
  where $V(t)$, the virtual time at time t, denotes the current round of service in the corresponding fluid fair queueing service.

- $F(p_i^k) = S(p_i^k) + L_i^k/r_i$
  where L_i^k is the length of the k^{th} packet of flow i.

The progression of the virtual time $V(t)$ is given by

$$\frac{dV(t)}{dt} = \frac{C(t)}{\sum_{i \in B(t)} r_i}$$

As a result of simulating fluid fair queueing, WFQ has the property that the worst case packet delay of a flow compared to the fluid service is upper bounded by one packet. A number of optimizations to WFQ, including closer approximations to the fluid service and reductions in the computational complexity have been proposed in literature (see [3] for an excellent survey).

2.4 Issues in Wireless Fair Queueing

From the description of fair queueing in wireline networks in Section 2.3 and the description of the channel characteristics in Section 2.2, it is clear that adapting wireline fair queueing to the wireless domain is not a trivial exercise. Specifically, wireless fair queueing needs to deal with the following unique issues.

- The failure of traditional wireline fair queueing in the presence of location-dependent channel error.
- The compensation model for flows that perceive channel error: how transparent should wireless channel errors be to the user?
- The trade off between full separation and compensation, and its impact on fairness of channel access.
- The trade-off between centralized versus distributed scheduling and medium access in a wireless cell.
- Limited knowledge at the base stations about uplink flows: how does the base station discover the backlogged state and arrival times of packets at the mobile host?
- Inaccuracies in monitoring and predicting the channel state, and its impact on the effectiveness of the compensation model.

We now address three of the issues listed above: why fluid fair queueing fails in wireless channels, what are the trade-offs between separation and compensation, and what state does the base station need with respect to uplink flows in order to schedule both uplink and downlink flows jointly.

2.4.1 Why Wireline Fair Queueing Fails Over Wireless Channels

Consider three backlogged flows during the time interval [0,2] with $r_1 = r_2 = r_3$. Flow 1 and flow 2 have error free channels while flow 3 perceives a channel error during the time interval [0,1). By applying equation (1) over the time periods [0,1) and [1,2], we arrive at the following channel capacity allocation:

$$W_1[0,1) = W_2[0,1) = 1/2, W_1[1,2] = W_2[1,2] = W_3[1,2] = 1/3.$$

Now, over the time window [0,2], the allocation is

$$W_1[0,2] = W_2[0,2] = 5/6, W_3[0,2] = 1/3.$$

which does not satisfy the fairness property of equation (1). This simple example illustrates the difficulty in defining fairness in a wireless network, even in an idealized model. In general, due to location-dependent channel errors, server allocations designed to be fair over one time interval may be inconsistent with fairness over a different time interval, though both time intervals have the same backlogged set.

In the fluid fair queueing model, when a flow has nothing to transmit during a time window $[t, t+\Delta]$, it is not allowed reclaim the channel capacity that would have been allocated to it during $[t, t+\Delta]$ if it were backlogged at t. However, in a wireless channel, it may happen that the flow is backlogged, but unable to transmit due to channel error. In such circumstances, should the flow be compensated at a later time? In other words, should channel error and empty queues be treated the same or differently? In particular, consider the scenario when flows f_1 and f_2 are both backlogged, but f_1 perceives a channel error while f_2 perceives a good channel. In this case, f_2 will additionally receive the share of the channel which would have been granted to f_1 in the error-free case. The question is whether the fairness model should readjust the service granted to f_1 and f_2 in a future time window in order to compensate f_1. The traditional fluid fair queueing model does not need to address this issue since in a wireline model, either all flows are permitted to transmit or none of them is.

In order to address this issue, the wireless fair queueing algorithms differentiate between a non-backlogged flow, and a backlogged flow that perceives channel error. A flow that is not backlogged does not get compensated for lost channel allocation. However, a backlogged flow f that perceives channel error is compensated in future, when it perceives a clean channel, and this compensation is provided at the expense of those flows that received additional channel allocation when f was unable to transmit. Of course, this compensation model makes channel errors transparent to the user to some extent, but only at the expense of separation of flows. In order to achieve a trade-off between compensation and separation, we bound the amount of compensation that a flow can receive at any time. Essentially, wireless fair queueing seeks to make short error bursts transparent to the user so that long-term throughput guarantees are ensured, but exposes prolonged error bursts to the user.

2.4.2 Separation versus Compensation

Exploring the trade-off between separation and compensation further, we illustrate a typical scenario and consider several possible compensation schemes. Let flows f_1, f_2, and f_3 be three flows that share a wireless channel, with equal weights. Let f_1 perceive a channel error during a time window $[0, 1)$, and during this time window, let f_2 receive all the additional channel allocation that was scheduled for f_1 (for example, because f_2 has packets to send at all times, while f_3 has packets to send only at the exact time intervals determined by its rate). Now, suppose that f_1 perceives a clean channel during $[1, 2]$. What should the channel allocation be?

During $[0, 1]$, the channel allocation was as follows:

$$W_1[0,1) = 0, W_2[0,1) = 2/3, W_3[0,1) = 1/3.$$

Thus, f_2 received $1/3$ units of additional channel allocation at the expense of f_1, while f_3 received exactly its contracted allocation. During $[1, 2]$, what should the channel allocation be? In particular, there are two questions that need to be answered.

- Is it acceptable for f_3 to be impacted due to the fact that f_1 is being compensated even though f_3 did not receive any additional bandwidth?
- Over what time period should f_1 be compensated for its loss?

In order to provide separation for flows that receive exactly their contracted channel allocation, flow f_3 should not be impacted at all by the compensation model. In other words, the compensation should only be between flows that lag their error-free service and flows that lead that error-free service, where the error-free service denotes the service that a flow would have received if all the channels were error-free.

The second question is how long it takes for a lagging flow to recover from its lag. Of course, a simple solution is to starve f_2 in $[1, 2]$ and allow f_1 to catch up with the following allocation:

$$W_1[1,2] = 2/3, W_2[1,2] = 0, W_3[1,2) = 1/3.$$

However, this may end up starving flows for long periods of time when a backlogged flow perceives channel error for a long time. Of course, we can bound the amount of compensation that a flow can receive, but that still does not prevent pathological cases in which a single backlogged among a large set of backlogged flows perceives a clean channel over a time window, and is then starved out for a long time till all the other lagging flows catch up. In particular, the compensation model must provide for a graceful degradation of service for leading flows while they give up their lead.

2.4.3 Incomplete State at the Base Station for Uplink Scheduling

In a cell, hosts are only guaranteed to be within the range of the base station and not other hosts, and all transmissions are either uplink or downlink. Thus, the base station is the only logical choice for the scheduling entity in a cell. While the base station has full knowledge of the current state of each downlink flow (i.e. whether it is backlogged, and the arrival times of the packets), it has limited and imperfect knowledge of the current state of each uplink flow. In particular, a base station may not know precisely when a previously non-backlogged flow becomes backlogged, and the precise arrival times of uplink packets in this case. The lack of such knowledge has an impact on the accuracy of scheduling and delay guarantees that can be provided in wireless fair queueing.

This problem can be alleviated in part by piggybacking flow state on uplink transmissions, but newly backlogged flows may still not be able to notify their state to the base station. For a backlogged flow, the base station only needs to know if the flow will continue to remain backlogged even after it is allocated the channel once. This information can be easily obtained by the base station by adding a one bit field in the packet header. For a non-backlogged flow, the base station needs to know precisely when the flow becomes backlogged. As far as we know, there exists no way to guarantee up-to-date flow state for uplink flows at the base station except for periodic polling, which may be wasteful in terms of consuming signaling bandwidth. In related work [6, 7], two alternative mechanisms are proposed for a base station to obtain this information, but these mechanisms do not guarantee that the base station will indeed obtain the precise state of uplink flows.

3 Policies and Mechanisms

In this section, we present a generic framework for wireless fair queueing, identify the key components of the framework, and discuss alternative policies and mechanisms for each of the components. The next section provides instantiations of these alternatives with specific wireless fair queueing algorithms from contemporary literature.

Wireless fair queueing involves the following five components:

- *Error-free service model*: defines an ideal fair service model assuming no channel errors. This is used as a reference model for channel allocation.
- *Lead and lag model*: determines which flows are leading or lagging their error free service, and by how much.
- *Compensation model*: compensates lagging flows that perceive an error-free channel at the expense of leading flows, and thus addresses the key issues of bursty and location-dependent channel error in wireless channel access.
- *Slot queue and packet queue decoupling*: allows for the support of both delay sensitive and error sensitive flows in a single framework and also decouples connection-level packet management policies from link-level packet scheduling policies.
- *Channel monitoring and prediction*: provides a (possibly inaccurate) measurement and estimation of the channel state at any time instant for each backlogged flow.

Figure 2 shows the generic framework for wireless fair queueing. The different components in the framework interact as follows: The error-free service is used as the reference model for the service that each flow should receive. Since a flow may perceive location-dependent channel error during any given time window, the lead and lag model specifies how much additional service the flow is eligible to receive in the future (or how much service the flow must relinquish in the future). The goal of wireless fair queueing is to use the compensation model in order to make short location-dependent error bursts transparent to the lagging flows while provide graceful service degradation for leading flows. In order to support both delay-sensitive and error-sensitive flows, the scheduler only allocates slots to flows and does not determine which packet will be transmitted when a flow is allocated a slot. Finally, the channel prediction model is used to determine whether a flow perceives a clean or dirty channel during each slot.

Once a flow is allocated a slot, it still needs to perform the wireless medium access algorithm in order to gain access to the channel and transmit a packet. In this article, we do not explore the interactions between the scheduling algorithm and the medium access algorithm.

3.1 Error-free Service Model

The error-free service model provides a reference for how much service a flow should receive in an ideal error-free channel environment. As mentioned above, the goal of wireless fair queueing is to approximate the error-free service model by making short error bursts transparent to a flow, and only expose prolonged channel error to the flow.

Most contemporary wireless fair queueing algorithms use well known wireline fair queueing algorithms for their error-free service model. Three choices have typically been used:

1. Wireline fair queueing algorithms such as WFQ [2] or PGPS [1], in which the rate of change of the virtual time (dV/dt) is explicitly simulated. Simulating the virtual time explicitly

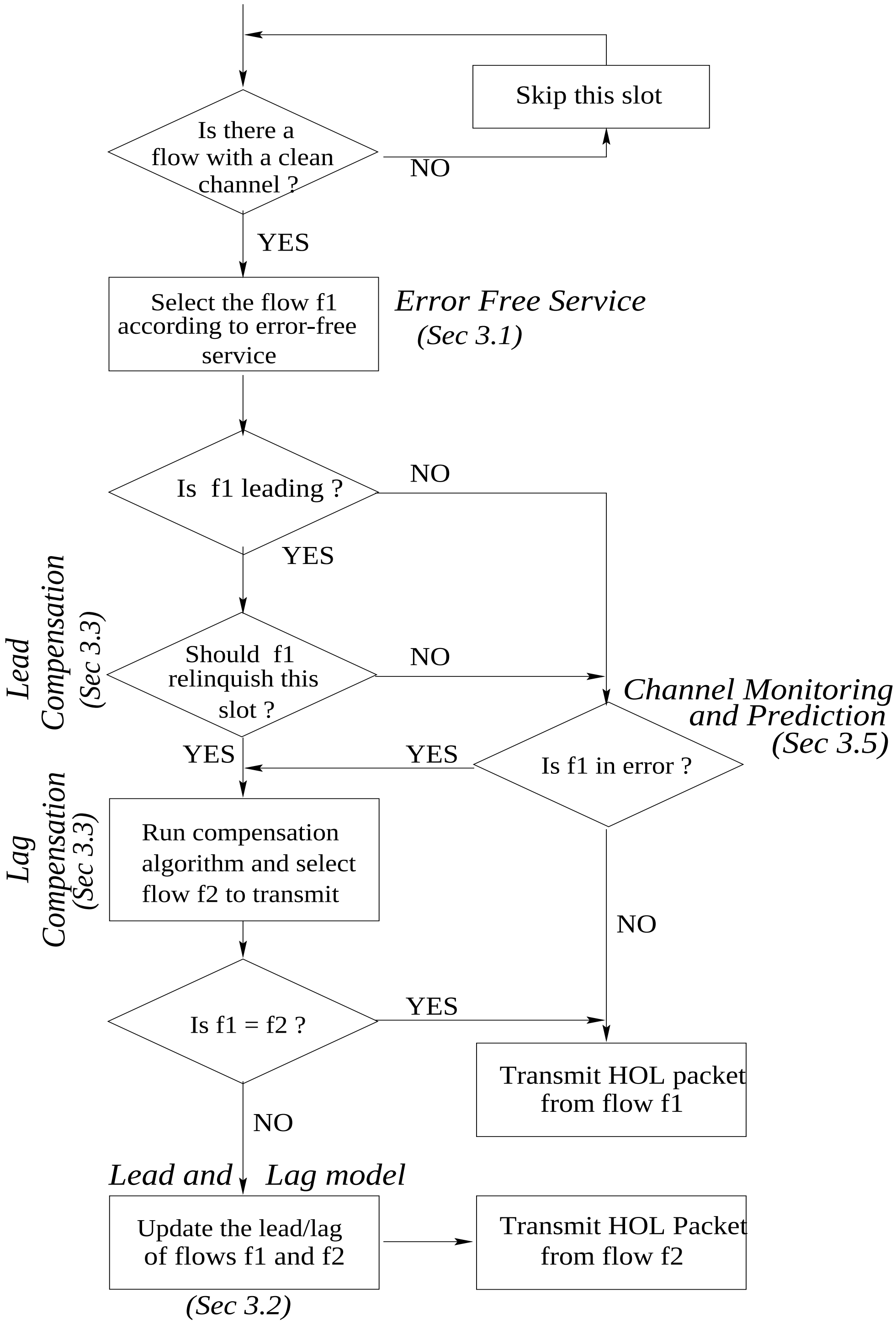

Figure 2: Generic Framework for Wireless Fair Queueing

When we run the lag compensation algorithm to select the alternate flow f_2, the algorithm picks a flow f_2 that has packets to send and perceives a clean channel at that instant.

can be computationally expensive. IWFQ [6] uses WFQ or WF2Q [4] to compute its error free service.

2. Wireline fair queueing algorithms such as STFQ [9], in which the virtual time is not explicitly simulated. In STFQ, for example, the virtual time is set to the start tag of the packet that is currently being served. CIF-Q [8] uses STFQ to compute its error free service.

3. A variation of fluid fair queueing which allows for a decoupling of delay and rate in the scheduler. This is achieved by allocating a rate weight r_i and a delay weight ϕ_i for each flow i, and modifying the tagging mechanism described in Section 2.3 as follows:

   - $S(p_i^k) = max\{V(A(p_i^k)), S(p_i^{k-1}) + L_i^{k-1}/r_i\}$
   - $F(p_i^k) = S(p_i^k) + L_i^k/\phi_i$

   This algorithm was proposed by WFS [7], and is revisited in Section 4.4.

3.2 Lead and Lag Model

In Section 2.1, we described the lag, lead, lagging flows, and leading flows in terms of the difference in service received by a flow in its real service compared to its idealized service. We now refine this definition: the *lag of a lagging flow* denotes the amount of additional service to which it is entitled in the future in order to compensate for lost service in the past, while the *lead of a leading flow* denotes the amount of additional service that the flow has to relinquish in the future in order to compensate for additional service received in the past.

There are two distinct approaches to computing lag and lead.

1. The lag of a flow is the difference between the error-free service and real service received by the flow. In this case, a flow that falls behind its error-free service is compensated irrespective of whether its lost slots were utilized by other flows. SBFA [10] uses this approach.

2. The lag of a flow is the number of slots allocated to the flow during which it could not transmit due to channel error, but another backlogged flow that had no channel error transmitted in its place and increased its lead. In this case, the lag of a flow is incremented upon a lost slot only if another flow that took this slot is prepared to relinquish a slot in the future. IWFQ [6], WFS [7], and CIF-Q [8] use this approach.

Lead and lag may be upper bounded by flow-specific parameters. An upper bound on lag is the maximum error burst that can be made transparent to the flow, while an upper bound on lead is the maximum number of slots which the flow must relinquish in the future in order to compensate for additional service received in the past.

3.3 Compensation Model

The compensation model is the key component of wireless fair queueing algorithms. It determines how lagging flows make up their lag and how leading flows give up their lead. Thus the compensation model has to address three main issues: (a) When does a leading flow relinquish the slots that are allocated to it? (b) When are slots allocated for compensating lagging flows? and (c) How are compensation slots allocated among lagging flows? We now explore the design choices for each issue.

Leading flows are required to give up some of the slots that are allocated to them in error-free service so that lagging flows can use these slots to reduce their lag. There are three possible choices for a leading flow to relinquish its lead.

- The first choice, adopted by IWFQ [6], is for a leading flow to relinquish all slots till it becomes in sync. The problem with this approach is that a leading flow that has accumulated a large lead because other flows perceive large error bursts may end up being starved of channel access at a later time when all lagging flows start to perceive clean channels.

- The second choice is for a leading flow to relinquish a fraction of the slots allocated to it. The fraction of slots relinquished may be constant, as in CIF-Q [8], or may be proportional to the lead of the flow, as in WFS [7]. The advantage of relinquishing a fraction of the allocated slots is that service degradation is graceful. In WFS, for example, the degradation in service decreases exponentially as the lead of a flow decreases.

- The third choice is for a leading flow to never relinquish its lead. In this case, we assume that there is a separate reserved portion of the channel bandwidth that is dedicated for the compensation of lagging flows. SBFA [10] uses this approach.

Lagging flows must receive additional slots in excess of their error-free service in order to make up for lost service in the past. We call these additional slots 'compensation slots'. There are three choices for allocating compensation slots to lagging flows:

- Compensation slots are preferentially allocated till there is no lagging flow that perceives a clean channel, as in IWFQ [6]. As a result, lagging flows have precedence in channel allocation over in sync and leading flows.

- Compensation slots are allocated only when leading flows relinquish slots, and in CIF-Q [8] and WFS [7].

- Compensation slots are allocated from a reserved fraction of the channel bandwidth that is set aside specifically to compensate lagging flows, as in SBFA [10].

Giving lagging flows precedence in channel allocation may disturb in sync flows and cause them to become lagging even if they perceive no channel error. On the other hand, allocating a separate reserved portion of the channel statically bounds the amount of compensation that can be granted. Thus, we believe that the second approach, i.e. explicitly swapping slots between leading and lagging flows, is best suited for achieving wireless fair service.

The final question in the compensation model is how to distribute compensation slots among lagging flows. Three design choices have been explored in contemporary algorithms:

- The lagging flow with the largest lag is allocated the compensation slot, as in CIF-Q [8].

- The history of when flows became lagging is maintained, and the flows are compensated according to the order in which they became backlogged, as in IWFQ [6] and SBFA [10].
- The lagging flows are compensated fairly, i.e. each lagging flow receives the number of compensation slots in proportion to its lag, as in WFS [7].

Among these options, fair compensation achieves the goal of short-term fairness in wireless fair service, but is computationally more expensive than the other two options.

3.4 Slot Queues and Packet Queues

Typically, packets are tagged as soon as they arrive in wireline fair queueing algorithms. This works well if we assume no channel error, i.e. a scheduled packet will always be transmitted and received successfully. However, in a wireless channel, packets may be corrupted due to channel error, and an unsuccessfully transmitted 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, one additional level of abstraction can be used in order to decouple 'slots', the unit of channel allocation, from 'packets', the unit of data transmission. When a packet arrives in the queue of a flow, a corresponding slot is generated in the slot queue of the flow, and tagged according to the wireless fair queueing algorithm. At each time, the scheduler determines which slot will get access to the channel, and the head-of-line packet in the corresponding flow queue is then transmitted. 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 enables the scheduler to support both error-sensitive flows and delay-sensitive flows according to the wireless fair service model. Error-sensitive flows will not delete the head-of-line packet upon channel error during transmission, but delay-sensitive flows may delete the head-of-line 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. Essentially, the approach is to limit the scope of the scheduler to determine only which flow is allocated the channel next, and let each flow make its own decision about which packet in the flow it wishes to transmit. In our scheduling model, we support any queueing and packet dropping policy at the flow level because we decouple slot queues from packet queues[2].

3.5 Channel monitoring and prediction

Perfect channel-dependent scheduling is only possible if the scheduler has accurate information about the channel state of each backlogged flow. The location-dependent nature of channel

[2] The slot queue and packet queue decoupling as described in this section is applicable for fixed size packets only. Adapting this mechanism for variable size packets involves solving several subtle issues, which are beyond the scope of this article.

error requires each backlogged flow to monitor its channel state continuously, based on which the flow may predict its future channel state and send this information to the scheduler.

Errors in the wireless channel typically occur over bursts and are highly correlated in successive slots, but possibly uncorrelated over longer time windows. Thus fairly accurate channel prediction can be achieved using an n-state Markov model. In fact, we have found that even using a simple *one step* prediction algorithm (predict slot $i+1$ is good if slot i is observed to be good, and bad otherwise) results in an acceptable first cut solution to this problem.

The algorithms discussed in this article do not make any assumptions about the exact error model, though they assume an upper bound on the number of errors during any time window of size T_i, i.e. flow i will not perceive more than e_i errors in any time window of size T_i, where e_i and T_i are per-flow parameters for flow i. The delay and throughput properties that are derived for the wireless fair queueing algorithms are typically 'channel-conditioned', i.e. conditioned on the fact that flow i perceives no more than e_i errors in any time window of size T_i [6, 7].

4 Algorithms for Wireless Fair Queueing

In the last section, we described the key components of a generic wireless fair queueing algorithm, and discussed possible design choices for each of the components. In this section, we briefly provide an overview of four contemporary wireless fair queueing algorithms, and compare their characteristics.

The four algorithms we describe are the Idealized Wireless Fair Queueing algorithm (IWFQ) [6], the Channel-condition Independent Fair Queueing algorithm (CIF-Q) [8], the Server Based Fairness Approach (SBFA) [10], and the Wireless Fair Service algorithm (WFS) [7].

4.1 Idealized Wireless Fair Queueing (IWFQ)

In IWFQ, the error-free service is simulated by WFQ [2] or WF2Q [4]. The start tag and finish tag of each slot are assigned as in WFQ. The *service tag* of a flow is set to the finish tag of its head of line slot. In order to schedule a transmission, IWFQ selects the flow with the minimum service tag among the backlogged flows that perceive a clean channel.

Each flow i has a lead bound of l_i and a lag bound of b_i. The service tag of flow i is not allowed to increase by more than l_i above, or decrease by more than b_i below, the service tag of its error free service.

The compensation model in IWFQ is implicit. If a flow perceives channel error, it retains its tag (and hence, precedence for transmission when it becomes error free). Likewise, if a flow receives additional service, its service tag increases. Consequently, lagging flows end up having lower service tags than flows that are in sync or leading, and hence have precedence in channel access when they become error free.

As a consequence of the compensation model in IWFQ, a flow that is lagging for a long time will be able to capture the channel once it becomes error free. Likewise, a leading flow may be starved out of channel access for long periods of time. Thus the compensation model in IWFQ does not support graceful degradation of service. Additionally, in sync flows may be affected during the compensation that is granted to lagging flows.

4.2 Channel-condition Independent Fair Queueing (CIF-Q)

In CIF-Q, the error free service is simulated by STFQ [9]. Each flow has a *lag* parameter that is positive if the flow is lagging and negative if it is leading.

When a lagging or in sync flow i is allocated the channel, if i perceives a clean channel, then it transmits a packet. Otherwise, if there is a backlogged flow j which perceives a clean channel and transmits instead of i, then the lag of i is incremented and the lag of j is decremented.

A leading flow i retains a fraction α of its service and relinquishes a fraction $1-\alpha$ of its service, where α is a system parameter that governs the service degradation of leading flows. When a leading flow relinquishes a slot, it is allocated to the lagging flow with the largest lag.

As a consequence of its compensation model, CIF-Q provides a graceful linear degradation in service for leading flows. Additionally, it performs compensation of lagging flows by explicitly swapping slots with leading flows, thus ensuring that in sync flows are not affected. CIF-Q thus overcomes two of the main drawbacks of IWFQ.

4.3 Server Based Fairness Approach (SBFA)

SBFA provides a framework in which different wireline scheduling algorithms can be adapted to the wireless domain. The error free service in SBFA is the desired wireline scheduling algorithm that needs to be adapted to the wireless domain.

SBFA statically reserves a fraction of the channel bandwidth for compensation. In contrast with other wireless fair queueing algorithms, SBFA tries to compensate the lagging flows using the reserved bandwidth rather than swapping slots between leading and lagging flows.

In SBFA, a virtual flow called *long term fairness server (LTFS)* is created to provide compensation and allocated a rate weight that is adjusted to reflect the bandwidth reservation for compensation. When a flow that is allocated a slot is unable to transmit, the slot is queued to the LTFS. The scheduling algorithm then treats LTFS on par with other packet flows for channel allocation.

There is no concept of leading flows in SBFA. The lag of a flow is not explicitly bounded, and the order of compensation among lagging flows is according to the order in which their slots are queued in the LTFS.

As a consequence of the compensation model, SBFA provides long term fairness, but not short-term fairness or worst case delay bounds for packet transmission. Specifically, the rate of compensation is bounded by the reserved portion of the channel bandwidth. In sync flows may be affected by the compensation. The service degradation for leading and in sync flows is graceful and the available service is lower bounded by the minimum rate contracts established for the flow.

4.4 Wireless Fair Service (WFS)

In WFS, the error free service is computed by the modified fair queueing algorithm described in Section 3.1 in order to achieve a delay-bandwidth decoupling in the scheduler. This decoupling expands the schedulable region for WFS. Unlike traditional fair queueing algorithms, WFS can support flows with high bandwidth and high delay requirements, as well as flows with low bandwidth and low delay requirements.

Each flow i has a lead bound of $l_i^{\max}$ and a lag bound of $b_i^{\max}$. A leading flow with a current lead of l_i relinquishes a fraction $l_i/l_i^{\max}$ of its slots, while a lagging flow with a current lag of b_i receives a fraction $b_i/\sum_{j\in S} b_j$ of all the relinquished slots, where S is the set of backlogged flows. Effectively, leading flows relinquish their slots in proportion to their lead, and relinquished slots are fairly distributed among lagging flows.

As a consequence of the compensation model, service degradation is graceful for leading flows and the fraction of slots relinquished by leading flows decreases exponentially. Compensation among lagging flows is fair. WFS achieves both short-term and long-term fairness, as well as delay and throughput bounds. The error-free service of WFS allows it to decouple delay and bandwidth requirements.

All the algorithms described in this section share several common features. First, they all specify an error-free service model and then try to approximate the error-free service even when some flows perceive location-dependent error, by implicitly or explicitly compensating lagging flows at the expense of leading flows. Second, they all have similar computational complexity. Third, they all provide mechanisms to bound the amount of compensation that can be provided to any flow, thereby controlling the amount of channel error that can be made transparent to error-prone flows. Finally, all of them try to achieve at least some subset of wireless fair service. Among the algorithms described, CIF-Q and WFS achieve all the properties of wireless fair service, and WFS additionally achieves delay-bandwidth decoupling.

5 Illustration of Wireless Fair Service

In this section, we illustrate different aspects of wireless fair service through simple examples. We have selected the WFS algorithm as a specific instantiation of wireless fair queueing algorithms, and used this algorithm in the examples.

As described in the previous sections, the key features of wireless fair service include separation between flows, 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. Additionally, WFS provides decoupling of delay and bandwidth. These features are illustrated in the examples described below.

The following are the performance measures used in the simulation: 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, i.e. worst case delay of the head of line packet of a freshly backlogged flow. 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 flows and Markov-modulated Poisson (MMPP) flows in our simulations. For the MMPP flows, 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 indicate that it is reasonably effective for typical wireless channel error models.

We present four examples in this section, where each one is used to demonstrate a specific feature. 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 flow can maintain its throughput, even when it drops packets due to channel error or delay violation.

Example 1: Decoupling of rate and delay in WFS. Consider three Poisson flows with error-free channels. Flow 1 has an average rate of 0.11, Flows 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 flows. This reduces the algorithm to the Wireless Fair Queueing algorithm [1, 2]. 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 flows are proportional to their weights, and the configuration is schedulable. [3]

(b) Now, we change the delay weights for each of the flows, 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 Flow 1, which has a larger delay weight than the other flows, experiences a much smaller delay, even though its rate is smaller than the other two flows. On the other hand, Flow 3 has a large rate, but it sees a large delay, as it has a smaller delay weight. Thus wireless fair service can support low rate, low delay flows, as well as high rate, high delay flows.

Example 2: Error-sensitive vs. Delay-sensitive flows. We now consider an example where the channel is error-prone, and where 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 for a maximum number of times and encounters a channel error on all its attempts.

We consider three flows, where Flows 1 and 2 are MMPP and the arrivals occur according to a Poisson process of rate 1.5 when the Markov chain is in the ON state, and Flow 3 is a constant flow with a rate of 0.25 (i.e packet inter-arrival time of 4). The channel for Flows 1 and 2

[3] 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	0.11	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: Flow parameters and results for Example 1: Scenario (a).

	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: Flow parameters and results for Example 1: Scenario (b).

evolve according to a two-state discrete Markov chain having a steady state error probability $P_E = 0.3$ with $p_g + p_e = 0.1$. Flow 3 has an error-free channel. The rate weights for all flows 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 the compensation model provides for short term fairness and throughput bounds even when the flows perceive short error bursts.

(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 flows 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 Flow 3, which has an error-free channel, does not experience a change in its throughput, due to the

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

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

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

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

compensation given to other flows. This illustrates the separation of flows.

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 while 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 gracefully converges to a rate to 0.33 as it reduces its lag to 0.

Example 4: Adaptive Flows 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 compensation. A flow can react to this packet dropping by generating additional packets equal

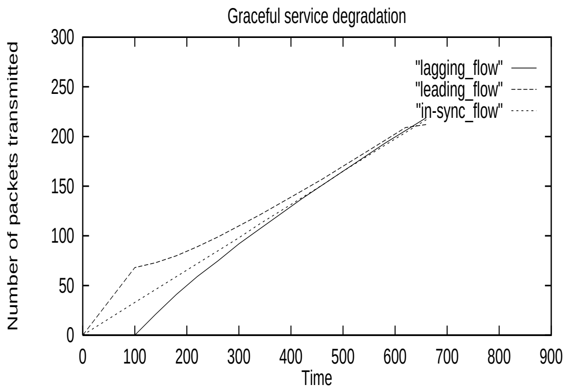

Figure 4: Graceful Service Degradation

	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

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 flow 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 flows 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.

6 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

<table>
<tr><th rowspan="2"></th><th rowspan="2">$D_{\max}$</th><th rowspan="2">Non-adaptive</th><th colspan="5">Adaptation window</th></tr>
<tr><th>100</th><th>50</th><th>40</th><th>30</th><th>10</th></tr>
<tr><td rowspan="3">W</td><td>∞</td><td>.3324</td><td colspan="5"></td></tr>
<tr><td>100</td><td>.3275</td><td>.3283</td><td>.3291</td><td>.3304</td><td>.3308</td><td>.3319</td></tr>
<tr><td>50</td><td>.3082</td><td>.3265</td><td>.3268</td><td>.3273</td><td>.3292</td><td>.3298</td></tr>
</table>

Table 6: Effect of adaptive nature of flow on throughput

for supporting bounded delay access to a shared uni-directional wireline link, wireline fair queueing algorithms cannot be applied directly to the wireless domain because of the unique characteristics of wireless channels, such as location-dependent and bursty channel error, channel contention, dynamic channel capacity, shared broadcast channels for uplink and downlink flows, etc.

In this article, we have identified some of the issues that need to be solved in order to achieve wireless fair queueing, and we have described some approaches to initial solutions in this area. Based on a preliminary evaluation of these approaches, we believe that currently proposed wireless fair queueing algorithms can effectively provide channel-conditioned guarantees on delay, and some degree of separation between flows.

While many important issues remain to be solved, we believe that there are two key areas that require more research: (a) the development of simple and accurate channel prediction models that can be easily implemented within the framework of popular medium access protocols, and (b) the development of simple and robust medium access protocols that enable wireless fair queueing, including swapping slots between flows based on channel error and the propagation of precise uplink state to the base station.

As mentioned earlier on in this article, wireless fair queueing is only one way to provide fair access to the scarce and shared wireless channel. Several other algorithms for such sharing have been proposed [5, 11]. A comparison of these algorithms with wireless fair queueing is another prime area for future research.

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] A. Demers, S. Keshav and S. Shenker, "Analysis and simulation of a fair queueing algorithm," *ACM SIGCOMM*, August 1989.

[3] H. Zhang, "Service disciplines for guaranteed performance service in packet-switching networks," *Proceedings of the IEEE*, 83(10), October 1995.

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

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

[7] S. Lu, T. Nandagopal, and V. Bharghavan, "Fair scheduling in wireless packet networks," *ACM MOBICOM*, October 1998.

[8] T.S. Ng, I. Stoica and H. Zhang, "Packet fair queueing algorithms for wireless networks with location-dependent errors," *IEEE INFOCOM*, March 1998.

[9] P. Goyal, H.M. Vin and H. Chen, "Start-time Fair Queueing: A scheduling algorithm for integrated service access," *ACM SIGCOMM*. August 1996.

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

[11] M. Srivastava, C. Fragouli, and V. Sivaraman, "Controlled Multimedia Wireless Link Sharing via Enhanced Class-Based Queueing with Channel-State-Dependent Packet Scheduling," *IEEE INFOCOM*, March 1998.