Wireless Networks 8, 231–247, 2002
© 2002 Kluwer Academic Publishers. Manufactured in The Netherlands.

A Unified Architecture for the Design and Evaluation of Wireless Fair Queueing Algorithms

THYAGARAJAN NANDAGOPAL
Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, USA

SONGWU LU
Department of Computer Science, University of California at Los Angeles, USA

VADUVUR BHARGHAVAN
Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, USA

Abstract. Fair queueing in the wireless domain poses significant challenges due to unique issues in the wireless channel such as location-dependent and bursty channel errors. In this paper, we present a *wireless fair service* model that captures the scheduling requirements of wireless scheduling algorithms, and present a *unified wireless fair queueing architecture* in which scheduling algorithms can be designed to achieve wireless fair service. We map seven recently proposed wireless fair scheduling algorithms to the unified architecture, and compare their properties through simulation and analysis. We conclude that some of these algorithms achieve the properties of wireless fair service including short-term and long-term fairness, short-term and long-term throughput bounds, and tight delay bounds for channel access.

Keywords: wireless scheduling, fair queueing, wireless networks, wireless fair service

1. Introduction

The growing use of wireless networks has brought the issue of providing fair wireless channel arbitration among contending flows to the fore. The wireless channel being a critical scarce resource, it is imperative to provide both short-term and long-term fairness in channel access since providing only best effort service can result in channel starvation for some contending stations for long periods of time. In wireline networks, *fluid fair queueing* has long been a popular paradigm for achieving instantaneous fairness and bounded delays in channel access. However, adapting wireline fair queueing algorithms to the wireless domain is non-trivial because of the unique problems in wireless channels such as location-dependent and bursty errors, channel contention, and joint scheduling of uplink and downlink flows in a wireless cell.

In the past few years, several wireless fair queueing algorithms have been developed [2,6–10], that provide varying degrees of short-term and long-term fairness, short-term and long-term throughput bounds, average case and worst case delay bounds, and graceful degradation for flows in the presence of channel error. However, there has not been any work to precisely characterize the desired service model in terms of a *wireless fair service*, and define a *unified wireless fair queueing architecture* to achieve wireless fair service. This is important for two reasons: (a) it provides a single framework in which to compare different wireless fair queueing algorithms and evaluate tradeoffs between these algorithms head-to-head, and (b) it serves as an architectural framework in which to develop new wireless scheduling algorithms. Given the emerging importance of wireless fair queueing and the diversity of contemporary wireless fair queueing algorithms proposed in the literature, we believe that such a study is overdue. To this end, this paper makes three contributions:

1. We present a *wireless fair service model* that captures the scheduling requirements in the wireless domain.
2. We present a *unified wireless fair queueing architecture* that serves as a framework to design wireless fair queueing algorithms. We then map seven recently developed wireless fair queueing algorithms onto this unified framework. These algorithms are: Channel State Dependent Packet Scheduling algorithm (CSDPS) [2], Idealized Wireless Fair Queueing algorithm (IWFQ) [6], Channel Independent Fair Queueing algorithm (CIF-Q) [8], Server Based Fairness algorithm (SBFA) [9], Wireless Fair Service algorithm (WFS) [10], a variant of IWFQ called Wireless Packet Scheduling algorithm (WPS) [6], and an enhancement of CSDPS that provides class based queueing (CBQ-CSDPS) [7].
3. We evaluate and compare the seven algorithms mentioned above via both simulation and analysis. Based on our evaluation, we conclude that two of these algorithms, WFS [10] and CIF-Q [8], achieve all properties of wireless fair service in the general case.

The rest of this paper is organized as follows. In section 2, we describe the channel model, the wireless fair service model, and the key issues in wireless fair queueing. In section 3, we present the unified architecture for wireless fair queueing. In section 4, we map the seven wireless fair queueing algorithms as instantiations of the generic architecture. In sections 5 and 6, we compare the algorithms through simulation and analysis. Section 7 concludes the paper.

2. Models and issues

2.1. Wireless channel model

We consider a packet cellular network, where each base station performs the scheduling of both uplink and downlink packet transmissions in its cell. All communication is constrained to be uplink or downlink. Neighboring cells are assumed to transmit on different logical channels. 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.

The key characteristics of the wireless channel include the following: (a) the *wireless 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) *mobile hosts do not have global channel state* (in terms of which other hosts contending for the same channel have packets to transmit, etc.), (e) the scheduling must take care of both *uplink and downlink flows*, and (f) mobile hosts are often constrained in terms of processing power and battery power. For simplicity, we assume that the packets are of the same size. The results presented in this paper can also be extended to variable packet sizes.

2.2. Service model

Fluid fair queueing has three important properties [3]: (a) fairness among backlogged flows even over infinitesimal time windows, (b) bounded delay channel access, and (c) guaranteed minimum throughput for backlogged flows. In summary, fluid fair queueing provides full *separation* between flows, i.e. the minimum guarantees provided for a flow are unaffected by the behavior of other flows. However, fluid fair queueing assumes that the channel is error-free, or at the very least, errors are not location dependent (i.e. all backlogged flows have the ability to transmit at a given time, or none of the flows can). Specifically, fluid fair queueing is neither fair nor able to provide minimum throughput bounds in the presence of location dependent channel error, as shown in section 2.3.

In order to capture the behavior of flows in a wireless environment while bearing the constraints of the channel in mind, 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. A flow that is neither leading nor lagging is said to be *in sync*.

In an effort to identify the requirements of flows in a channel-constrained wireless environment, we define a *wireless fair service model* for fair queueing in wireless channels with the following properties:

1. *Short-term fairness* among *in sync* backlogged flows that perceive a clean channel.
2. *Short-term throughput bounds* for flows with clean channels.
3. *Channel conditioned delay bounds* for packets.
4. *Long-term fairness* among backlogged flows with bounded channel error.
5. *Long-term throughput bounds* for all flows with bounded channel error.
6. Support for both *delay sensitive and error sensitive* data flows.
7. Optionally, *optimization of the schedulable region* by decoupling the delay and bandwidth requirements of flows.

Property 1 ensures that channel allocation is fair among backlogged flows that are in conformance with their error-free service and that are able to transmit packets. Property 2 further specifies that even if a flow has received additional service in a previous time window, its degradation of service in any subsequent time window must be graceful, i.e. a flow that has received excess service in the past must not be starved of channel access at any time in the future. The delay bound requirement of property 3 is subject to the fact that channel error is bounded for any flow over some time period, i.e. each flow i observes at most e_i errors in any time window of length T_i, where e_i and T_i are flow-specific parameters. Property 3 specifies that so long as a flow has bounded channel error, none of its packets must wait indefinitely to be served. Property 4 further stipulates that long term fairness is not violated so long as every backlogged flow has sufficient a number of error-free slots during which it can transmit its packets. Property 6 is very useful for handling both delay sensitive and error-sensitive flows in error-prone channels.

2.3. Issues in wireless fair queueing

In adapting fluid fair queueing to the wireless domain, three critical issues need to be addressed:

1. The failure of traditional fluid fair queueing in the presence of location-dependent channel error.
2. The compensation model for flows that perceive channel error: how transparent should wireless channel errors be to the user?
3. The tradeoff between full separation and compensation, and its impact on fairness of channel access.

In addition to these issues, other issues that are important are (a) handling inaccuracies in channel state prediction, (b) discovery of uplink flow state by the base station, and (c) coordination of scheduling and medium access. We briefly discuss (a) in the next section, however, (b) and (c) are beyond the scope of this paper.

We now explore the three issues listed above. In fluid fair queueing, each flow i is given a weight r_i, and 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:

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

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 $[0, 1)$. If the scheduler is aware of the channel state of flows, then it will not consider f_3 during $[0, 1)$. Hence, 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, service allocations that are 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.

The problem is that wireless fair queueing must distinguish between a non-backlogged flow (for which no compensation is provided in fair queueing) from a backlogged flow that perceives channel error. However, compensating for the latter will void the separation property of fair queueing. Exploring the tradeoff between separation and compensation further, consider in the above example that during the time window $[0, 1)$, f_1's offered load was only 1/3, while f_2 could use all the additional channel allocation. Thus, over $[0, 1)$, the channel allocation is $W_1[0, 1) = 1/3$, $W_2[0, 1) = 2/3$, $W_3[0, 1) = 0$, i.e. f_2 received 1/3 units of additional channel allocation at the expense of f_3, while f_1 received exactly its contracted allocation. During $[1, 2]$, what should the channel allocation be? In particular, there are three questions that need to be answered: (a) is it acceptable for f_1 to be impacted due to the fact that f_3 is being compensated even though f_1 did not receive any additional bandwidth? (b) over what time period should f_3 be compensated for its loss? and (c) should f_2 give up its excess channel allocation, and over what time period? These three issues are central to wireless fair queueing and are discussed in the next section.

3. Unified wireless fair queueing framework

The basic goal of wireless fair queueing algorithms is to emulate fluid fair queueing when all flows perceive error-free channels, but *swap* channel allocation between flows that perceive channel error and flows that perceive a clean channel in order to make short location-dependent error bursts transparent to the end user at the expense of providing coarser properties for delay, instantaneous fairness, and throughput. The wireless fair queueing algorithms considered in this paper differ in terms of how the swapping takes place, between which flows the swapping takes place, and how the compensation model is structured. However, all these algorithms can be thought of as instances of a unified wireless fair queueing architecture, which consists of the following five components:

- The *error-free service*, which defines an ideal fair service model assuming no channel errors.
- The *lead and lag model* in wireless service, which determines which flows are leading or lagging their error free service, and by how much.
- The *compensation model*, which 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 queues and packet queues*, which allow 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*, which provides a reliable and accurate measurement and estimation of the channel state at any time instant for each backlogged flow.

Figure 1 describes the interactions between these components in the unified architecture. Within the context of this architecture, a wireless fair queueing algorithm has the ability to "plug-in" different algorithms for each component. We now describe the components, and consider some popular algorithmic choices for each component.

3.1. Error-free service model

The error-free service provides a reference for how much service a flow should receive in an ideal error-free channel environment. Typically, the error-free service is some packetized approximation of fluid fair queueing. We briefly describe Weighted Fair Queueing [3], the error-free service model desired for the algorithms in this paper. Other choices include STFQ [4], Weighted Round Robin, WRR with WFQ-like spreading, and an enhanced fluid fair model that allows for delay-bandwidth decoupling [10].

In WFQ, each flow i in a set of flows F is allocated a rate weight r_i. The kth packet p_i^k of flow i is assigned a start tag $S(p_i^k)$ and a finish tag $F(p_i^k)$, 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 kth 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.
- $F(p_i^k) = S(p_i^k) + L_i^k/r_i$.
- $\mathrm{d}V/\mathrm{d}t = C(t)/\sum_{i \in B(t)} r_i$, where $B(t)$ is the set of backlogged flows at time t and $C(t)$ is the instantaneous channel capacity at time t.

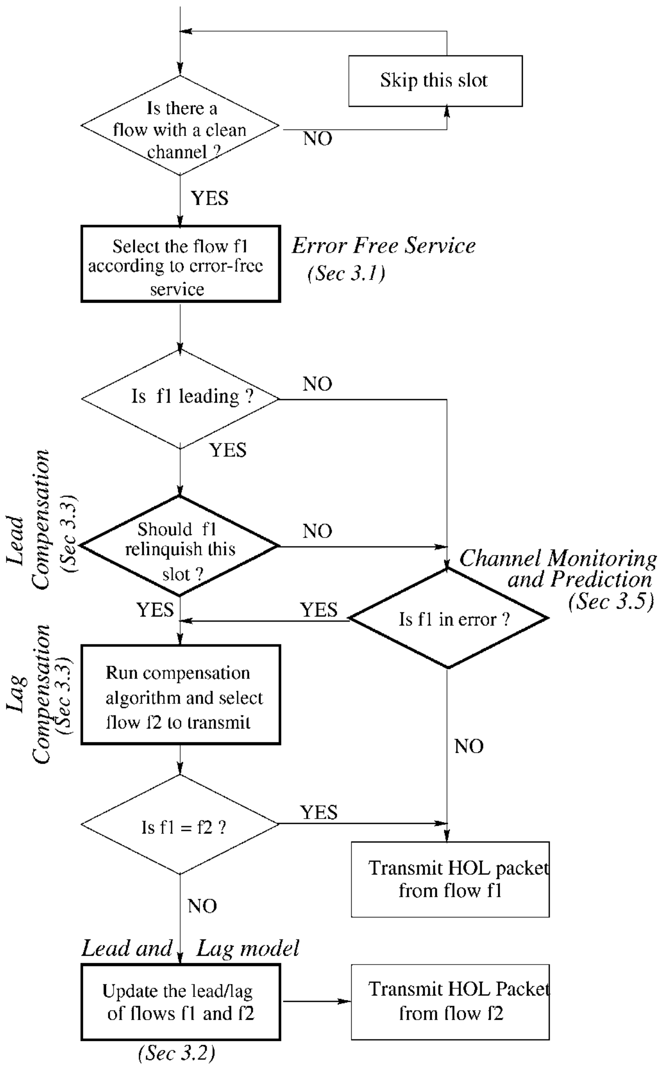
Skip this slot
Is there a flow with a clean channel ?
NO
YES
Select the flow f1 according to error-free service
Error Free Service (Sec 3.1)
Is f1 leading ?
NO
YES
Lead Compensation (Sec 3.3)
Should f1 relinquish this slot ?
NO
Channel Monitoring and Prediction (Sec 3.5)
YES
YES
Is f1 in error ?
Lag Compensation (Sec 3.3)
Run compensation algorithm and select flow f2 to transmit
NO
Is f1 = f2 ?
YES
Transmit HOL packet from flow f1
NO
Lead and Lag model
Update the lead/lag of flows f1 and f2
Transmit HOL Packet from flow f2
(Sec 3.2)

Figure 1. Interaction of components in the unified wireless fair queueing architecture. The bold boxes indicate programmable components. In the lag compensation box, the alternate flow f_2 is constrained to be a backlogged flow that perceives a clean channel.

- At each time, the packet with the minimum finish tag (i.e. the packet whose last bit would complete transmission first among all backlogged packets in the fluid model) is transmitted.

The version of WRR used by WPS, and STFQ are basically approximations of WFQ, that do not need to simulate the fluid model by computing $\mathrm{d}V/\mathrm{d}t$. We describe the error-free service of WFS in section 4.7.

3.2. Lead and lag model

Refining the notion of lead and lag introduced in section 2.2, 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. The set of leading flows, lagging flows and in sync flows may change dynamically over time.

There are two distinct approaches for computing lag and lead.

1. The lag of a flow is computed to be 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. This approach is used by IWFQ, CIF-Q, and SBFA.
2. The lag of a flow is computed to be 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. This approach is used by WFS and WPS.

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 purpose of the compensation component is to enable the lagging flows to reclaim service lost due to channel error, and to cause the leading flows to relinquish excess service received in the past. There are several possible compensation models for leading and lagging flows.

- *No explicit compensation.* A lagging flow is not compensated explicitly. Rather, the scheduling proceeds according to the error-free service, except that a flow perceiving channel error is skipped. So long as the offered load to the scheduler is stable (e.g., input traffic into each scheduler is policed), this approach provides long-term fairness among flows with bounded channel error. CSDPS uses this approach.
- *Flow with maximum lag is preferentially allocated the channel.* There are two variants to this compensation model: (a) the flow with the maximum lag is granted access to the channel whenever it can transmit (there is no explicit punishment of leading flows), and (b) the scheduler grants channel access to the flow with the minimum finish tag that perceives a clean channel. This mechanism explicitly maintains the precedence of lagging and leading flows, but in sync flows may also be affected due to compensation of lagging flows. IWFQ and CBQ-CSDPS use variants of this approach.
- *Leading and lagging flows swap slots.* There are several variants to this compensation model. When a leading flow is allocated a slot, it decides whether to relinquish or retain the slot according to one of three heuristics: (a) a leading flow always gives up its slots, (b) a leading flow gives up

a constant fraction of its slots (i.e. the compensation is linear), and (c) a leading flow gives up a varying fraction of its slots, where the fraction of slots relinquished decreases exponentially as the size of the lead decreases.

When a leading flow relinquishes a slot, a lagging flow is picked up according to one of three heuristics: (a) the lagging flow with the minimum finish tag, (b) the lagging flow with the maximum lag, and (c) a lagging flow from a weighted round robin allocation of lagging flows where the weight of a flow is its lag. The heuristic for the leading flow to relinquish its slots determines how gracefully leading flows degrade, while the heuristic for the lagging flow chosen for compensation determines how fairly lagging flows make up their lag. WFS and CIF-Q use variants of this approach.

- *Bandwidth is reserved for compensation.* A fraction of channel bandwidth is statically reserved for compensation by creating a "compensation flow" and scheduling it in the error-free service along with other flows. A lagging flow reclaims additional channel access from the slots allocated to the compensation flow. SBFA uses this approach.

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 never be lost. However, in a wireless channel, a lost packet may need to be retransmitted for an error-sensitive flow. Retagging the packet after a transmission loss 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. While the above description applies to the case of fixed packet sizes, the same concept can be extended to variable packet sizes also.

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.

3.5. Channel monitoring and prediction

Perfect channel-dependent scheduling is possible only if the scheduler has accurate information about the channel state for each backlogged flow. The location-dependent nature of channel 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 [11]. Thus, fairly accurate channel prediction can be achieved using an n-state Markov model [12]. 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 [6]. In general, the performance improves with the accuracy of the channel prediction algorithm.

It is important to note that for the purposes of this work, we make no assumptions about the exact channel error model, except for 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 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.

4. Instantiations of the unified wireless fair queueing architecture

The programmable components of the unified wireless fair queueing architecture that we consider are the error-free service algorithm, the lead/lag model, and compensation algorithm. In this section, we map seven wireless fair queueing algorithms onto the unified architecture. In the next two sections, we provide a comparative evaluation of these algorithms through simulation and analysis. We use the slot/packet decoupling mechanism used in [6], and one-step channel prediction for the last two components. These two components are orthogonal to the first three, and a specific choice of these components does not significantly impact the relative performance of the algorithms.

4.1. Channel State Dependent Packet Scheduling

Error-free service. CSDPS allows for the use of any error-free scheduling discipline. A typical example cited in the

CSDPS paper [2] is the standard weighted round robin algorithm (as opposed to the WRR with spreading in WPS – see section 4.3).

Lead and lag model. When a flow i is allocated a slot according to the error-free service, if flow i perceives a channel error, then CSDPS skips flow i and allocates the slot to the next flow according to the error-free service. In effect, CSDPS performs weighted round-robin among flows that perceive clean channels. As a result, CSDPS does not measure lag or lead for flows.

Compensation model. Since there is no concept of lag or lead, there is no compensation in CSDPS. As a consequence, a lagging flow can only make up its lag over the long term if leading flows cease to become backlogged sometime. Thus, CSDPS assumes that the input traffic is policed, and that the policing mechanism enforces stability. This is a limitation of CSDPS.

Implementation complexity. The implementation complexity of error-free service in CSDPS is low because of the use of WRR. However, WRR needs to check if each selected flow is backlogged and perceives a clean channel. This results in $O(n)$ pointer traversals for n flows (in WRR without spreading).

Impact and limitations. The service lost by a flow due to channel error is given to the next eligible flow, irrespective of whether that flow has received excess service or not. Thus, in-sync flows get disturbed, and receive service in excess of their error-free service. Since there is no compensation, CSDPS does not provide long-term and short-term fairness guarantees. However, for flows with error-free channels it can provide throughput guarantees. CSDPS by itself does not have any mechanism to commit a specific fraction of the available bandwidth to a flow, and it does not have a mechanism to enforce the allocations provided by the error-free service for flows that perceive channel error. This can result in misbehaving flows getting more than their fair share while other flows suffer.

4.2. Idealized Wireless Fair Queueing

Error-free service. IWFQ uses WFQ [6] for its error-free service according to the algorithm described in section 3.1.

Lead and lag model. Each arriving packet is tagged as in WFQ, and the service tag for a flow is set to the finish tag of its head-of-line packet. Among the flows that can transmit, i.e. backlogged flows with a clean channel, the flow with the least service tag is picked, and the head of line packet is transmitted.

IWFQ also simulates error-free service for identical arrivals. The lead of a leading flow is the difference between the service tag of the flow and the service tag of the flow in the error-free simulation, upper bounded by a per-flow parameter. The lag of a lagging flow is the difference between the service tag of the flow in the error-free simulation and the service tag of the flow in the real system, upper bounded by $B \cdot r_i$, where B is a scheduler parameter and r_i is the normalized weight of the flow (i.e. $\sum_{i\in F} r_i = 1$).

Compensation model. The compensation model implicitly favors channel access for lagging flows. Since precedence of tags is maintained, a lagging flow has a low service tag and captures the channel whenever it perceives a clean channel. Among lagging flows with clean channels, the flow with the lowest tag gets to transmit until it either perceives a dirty channel or its finish tag is greater than that of some other flow with a clean channel. This compensation model guarantees that lagging flows will catch up their lag, but may starve out leading flows in the short term.

Implementation complexity. The amortized cost for inserting a slot in sorted order is $O(\log n)$ for n flows. Computing the rate of increase of virtual time in WFQ takes $O(n)$ time, although algorithms such as STFQ [4] and SCFQ [5] eliminate this requirement. Searching for the backlogged flow with a clean channel with minimum service tag has a time complexity of $O(n)$ pointer traversals for n flows.

Impact and limitations. IWFQ was the first algorithm to propose a structured adaptation of fair queueing to the wireless domain.

Short-term fairness and throughput bounds in IWFQ are coarse because of the property that a lagging flow that starts to perceive a clean channel may capture the channel while its finish tag is minimum among flows with a clean channel. For the same reason, in sync flows may become lagging. However, IWFQ provides long-term fairness and bounded delay channel access.

4.3. Wireless Packet Scheduling

Error-free service. WPS uses WRR *with spreading of slots as in WFQ* as its error-free service. Consider three flows f_1, f_2, f_3 with weights of 0.2, 0.3, and 0.5, respectively. While the standard WRR would allocate slots according to the schedule:

$$\langle f_1, f_1, f_2, f_2, f_2, f_3, f_3, f_3, f_3, f_3\rangle,$$

WRR with spreading allocates slots according to the schedule

$$\langle f_3, f_2, f_3, f_1, f_3, f_2, f_3, f_1, f_2, f_3\rangle,$$

which is identical to the schedule generated by WFQ if all flows are backlogged. The mechanism to achieve this spreading is described in [6].

Lead and lag model. WPS generates a "frame" of slot allocation from the WRR-spreading algorithm. In each slot of the frame, if the flow that is allocated the slot is backlogged but perceives a channel error, then WPS tries to *swap* the slot with a future slot allocation within the same frame. If this is not possible (i.e. there is no backlogged flow perceiving a clean channel with a slot allocation later in the frame), then WPS increments the lag of the flow if another flow can transmit in its place (i.e. there is a backlogged flow with clean channel, but has been served its slot allocations for this frame), and the lead of this new alternate flow is incremented, where lead is negative lag. At the start of a frame, WPS computes the effective weight of a flow equal to the sum of its default weight and its lag, and resets the lag to 0. The frame is then generated based on the effective weights of flows.

Compensation model. The compensation is twofold in WPS. Intra-frame swapping is first attempted to compensate flows that encounter channel error by locally trading slot allocations. If this fails, the lag/lead accounting mechanism described above maintains the difference between the real service and the error-free service across frames. By changing the effective weight in each frame depending on the result of the previous frame, WPS tries to provide additional service to lagging flows at the expense of leading flows. In the ideal case, in sync flows are unaffected at the granularity of frames, though their slot allocations may change within the frame.

Implementation complexity. The implementation complexity of WRR-spreading is $O(n)$ for n flows. The worst case time complexity for intra-frame swapping is $O(n)$ pointer traversals for n flows.

Impact and limitations. WPS has performance characteristics similar to IWFQ. Thus, it has coarse short-term fairness and throughput bounds, but provides bounded delay channel access and long-term fairness. It may disturb in sync flows when intra-frame swapping fails to find a compensation flow. It is susceptible to a lagging flow accumulating a large lag. However, it prevents complete channel capture because each flow receives the effective weight worth of slots in each frame.

4.4. Channel-condition Independent Fair Queueing

Error-free service. CIF-Q uses Start Time Fair Queueing (STFQ) [4] as the error-free service. STFQ is an approximation of WFQ that eliminates the $\mathrm{d}V/\mathrm{d}t$ computation complexity by setting $V(t)$ to the start tag of the transmitting packet.

Lead and lag model. As in IWFQ, CIF-Q simulates an error-free service. The lag of a flow is the difference in service between the error-free service and the real service (i.e. lead is negative lag). A flow is considered to be "active" if it is either leading or backlogged. The error-free service is applied among all active flows. If a backlogged leading flow is allocated a slot, it relinquishes the slot with a probability of α, a system parameter. If an non-backlogged leading flow is allocated a slot, it relinquishes the slot. A relinquished slot is allocated to the lagging flow with the maximum normalized lag.

Compensation model. Lagging flows receive additional service only when leading flows relinquish slots. These relinquished slots are given to the lagging flow with the maximum normalized lag, where the normalization is done using the rate weight of a flow. As a result of this compensation policy, in sync flows are not disturbed if lagging flows can receive the additional service, and leading flows degrade their service gracefully. However, in pathological cases, a lagging flow may capture the channel, as in IWFQ, and starve out other flows.

Implementation complexity. The amortized time complexity for STFQ to insert service tags in sorted order (as in WFQ) is $O(\log n)$ for n flows. The complexity to compute virtual time is $O(1)$ in STFQ. In the event of a slot being allocated to a flow perceiving an error channel, the time complexity to find another flow to transmit in its place is $O(\log n)$.

Impact and limitations. CIF-Q can provide short-term and long-term fairness, and bounded delay channel access. Service degradation for leading flows is linear. Additional service for lagging flows is not short-term fair. In sync flows may be disturbed during redistribution of channel allocations that cannot be used by lagging flows or the selected flow. In the general case, CIF-Q achieves the properties of wireless fair service except that it disturbs in-sync flows, and in some pathological cases, a lagging flow may capture the channel as in IWFQ.

4.5. Enhanced Class Based Queueing with Channel State Dependent Packet Scheduling

Error-free service. CBQ-CSDPS combines a modified version of Class Based Queueing (CBQ) [14] with Channel State Dependent Packet Scheduling (CSDPS).

Lead and lag model. Rather than basing the lead/lag on the error-free service, CBQ-CSDPS maintains lead and lag based on the actual number of bytes s transmitted during each time window. A flow with a normalized weight r_i is leading if it has received channel allocation in excess of $s \cdot r_i$, and lagging if it has received channel allocation less than $s \cdot r_i$. Lagging flows are allowed precedence in transmission in order to make up their lag.

Compensation model. The compensation model of CBQ-CSDPS is similar to IWFQ in that lagging flows are given explicit precedence in channel access. This results in worst case behavior of channel capture by a lagging flow that starts to perceive a clean channel. Thus, short term fairness is not provided and the worst case delay bounds are coarse. Additionally, leads and lags are computed with respect to a time window of measurement; the properties of CBQ-CSDPS are sensitive to the time window of measurement.

Implementation complexity. If the error free service of CSDPS is WRR, the implementation complexity of CBQ-CSDPS follows the same arguments as CSDPS in section 4.1.

Impact and limitations. Short-term fairness is not provided. In sync flows are affected and leading flows may be starved of channel access, i.e. service degradation is not graceful in the worst case. CBQ-CSDPS can provide long term fairness and throughput bounds.

4.6. Server Based Fairness Approach

Error-free service. SBFA provides a generic framework for adapting different service disciplines to the wireless domain, though the properties satisfied by the service discipline in the wireline domain may not be translated to the wireless domain.

Lead and lag model. SBFA reserves a fraction of the channel bandwidth statically for compensation by specifying a *virtual compensation flow*. If a backlogged flow is allocated a slot but cannot transmit due to channel error, it enqueues a slot request in the compensation flow. The error-free service serves the compensation flow along with the other packet flows. When the compensation flow is allocated a slot, it turns over the slot to the flow to which its head-of-line slot request belongs. SBFA does not have the concept of a leading flow. The lag of a lagging flow is the number of slot requests in the compensation flow.

Compensation model. Since the compensation flow is treated like any other flow by the error-free service, in sync flows are not affected. However, when there are no slots in the compensation flow, its bandwidth is shared by all flows perceiving clean channels at that time instant. Thus, in sync flows receive excess service in this scenario. Lagging flows share the compensation flow; hence the rate of aggregate compensation received is statically bounded by the reserved share of the compensation flow. Head of line blocking of the compensation flow is not prevented. Leading flows do not give up their lead, since the lead of a leading flow is not monitored. SBFA is fundamentally different from the other algorithms discussed in this paper because it statically reserves a fraction of the channel for compensation. Thus, all the bounds supported by SBFA are only with respect to the remaining fraction of the channel bandwidth. The performance of SBFA is sensitive to the statically reserved fraction.

Implementation complexity. The performance of SBFA is dependent on the choice of the error-free service. For the compensation component of SBFA, the implementation complexity is a constant. This is because SBFA either transmits the slot chosen by the error-free service, or a replacement slot from the compensation flow, irrespective of channel state. The downside of this approach is that the worst case throughput bound of SBFA is extremely coarse.

Impact and limitations. SBFA provides long term fairness and throughput bounds for error-free flows. However, it does not provide short-term fairness or throughput bounds, and provides very coarse worst case delay bounds. Leading flows do not give up their lead, and lagging flows make up their lag from the reserved fraction of the channel. A lagging flow may capture compensation slots till it becomes in sync in the worst case. SBFA is sensitive to the reserved fraction parameter. If this value is less than the lag of flows over some time window, then error-prone flows cannot be guaranteed long-term fairness or throughput bounds.

4.7. Wireless Fair Service algorithm

Error-free service. WFS uses an enhanced version of WFQ in order to support delay-bandwidth decoupling. In WFS, each flow is allocated two parameters, a rate weight r_i and a delay weight ϕ_i. The start tag of a packet is computed as in WFQ, i.e. $S(p_i^k) = \max\{V(A(p_i^k)), S(p_i^{k-1}) + L_i^{k-1}/r_i\}$. However, the finish tag is computed based on ϕ_i rather than r_i, i.e. $F(p_i^k) = S(p_i^k) + L_i^k/\phi_i$. The service tag of a flow is the finish tag of its head-of-line packet. At a time t with virtual time $V(t)$, WFS transmits the flow with the minimum service tag and a clean channel subject to the constraint that the start tag of the head-of-line packet for the flow must be less than $V(t) + \varrho$, where ϱ is a *lookahead* parameter of the scheduler. If $\varrho = \infty$, the error-free service is earliest deadline first. If $\varrho = \infty$ and $r_i = \phi_i$, the error-free service is WFQ. If $\varrho = 0$ and $r_i = \phi_i$, the error-free service is WF^2Q [1]. Decoupling the delay and rate weights allows for delay-bandwidth decoupling.

Lead and lag model. If a backlogged flow perceiving an error channel is allocated the channel, its lag is increased only if there is another flow that can transmit in its place and increase its lead (or reduce its lag). Both lead and lag are bounded by per-flow parameters. In effect, the lag of a flow reflects the number of slots which the flow is entitled to make up in the future, and the lead of a flow reflects the number of slots it must relinquish in the future.

Compensation model. A leading flow with a lead of l and a lead bound of $l_{\max}$ relinquishes a fraction $l/l_{\max}$ of the slots allocated to it by error-free service. This leads to an exponential reduction in the number of slots relinquished as a function of the lead of the flow, and implies that a leading flow asymptotically relinquishes all its lead. WFS maintains a WPS-like "WRR with spreading" mechanism for determining which lagging flow will receive a relinquished slot. The weight of a lagging flow in the WRR is equal to its lag. As a result of this compensation model, compensation slots are fairly allocated among lagging flows, and service degradation is graceful for leading flows. In sync flows are not affected.

Implementation complexity. The error-free service has a time complexity of $O(\log n)$ for inserting service tags in sorted order as in all fair queueing algorithms. Traversing the WRR in order to determine the first available lagging flow to transmit a compensation slot has a time complexity of $O(n)$ pointer traversals.

Impact and limitations. WFS achieves the tightest short-term fairness and throughput bounds among all the algorithms considered in this paper. It achieves long-term fairness and throughput bounds, delay bounded channel access, and graceful degradation of leading flows. WFS satisfies the properties of wireless fair service. Additionally, it also has the optimal schedulable region because of delay–bandwidth decoupling.

5. Simulation results

In this section, we compare the algorithms in terms of the properties of wireless fair service. Specifically, we evaluate the performance of each algorithm by considering the following features: separation between flows, decoupling of rate and delay, size of the 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.

We have not presented CBQ-CSDPS in this version of the paper since it is ongoing work. We expect CBQ-CSDPS to perform similar to IWFQ.

5.1. Simulation environment

The following performance measures are used in the evaluation:

- 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. the maximum delay experienced by the head-of-line packet of a newly 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 40 simulation runs. To obtain measurements over short time windows, we measured the parameters over 10 different time windows, of size 200 time units each, in a single simulation run, and averaged the values obtained over 5 distinct simulation runs.

We have considered CBR sources, 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 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. Let p_{g} be the probability that the next time slot is good given that the current time slot is in error, and p_{e} be the probability that the next time slot is in error given that the current slot is good. Then, the steady-state probabilities P_{G} and P_{E} of being in the good and bad states, respectively, are given by

$$P_{\mathrm{G}} = \frac{p_{\mathrm{g}}}{p_{\mathrm{g}} + p_{\mathrm{e}}} \quad \text{and} \quad P_{\mathrm{E}} = \frac{p_{\mathrm{e}}}{p_{\mathrm{g}} + p_{\mathrm{e}}}.$$

We also consider bursty error models, in which the error burst lengths are uniformly distributed. For the channel prediction algorithm, we use one-step predictions, 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.

For the IWFQ [6] simulations, we do not bound the maximum credits and debits allowed for a flow. For CIF-Q, unless explicitly mentioned, we set $\alpha = 0.5$. For WFS [10] simulations, we set $\varrho = \infty$ and $\phi_i = r_i$ unless explicitly mentioned otherwise. For SBFA, we set the compensation fraction to 0.2. We have not simulated CBQ-CSDPS in this work.

We present six examples in this section. Example 1 illustrates the error-free service model, and the delay-bandwidth decoupling in WFS. Example 2 shows the performance of error-sensitive and delay-sensitive flows. Example 3 illustrates the service degradation property for leading flows. Example 4 illustrates the importance of a good channel prediction mechanism. Example 5 shows a particular case when all the algorithms perform similarly. Example 6 shows how an adaptive source can improve its throughput by adapting to packet drops due to channel error or delay-violation.

Example 1. Error-free service

In this example, we show that in the error-free case, each algorithm performs according to its error-free service model.

a. Consider three Poisson sources with error-free channels. Source 1 has an average rate of 0.111, sources 2 and 3 have average rates of 0.444 each.

The simulation results for flows 1 and 2 are given in table 1. As expected, the rates obtained by the sources are proportional to their weight, and the configuration is schedulable.

Table 1
Results for example 1(a): flows 1 and 2.

Algorithm	Flow 1						Flow 2					
	W	P_{l}	$D_{\max}$	D_{avg}	σ_D	d^{nq}	W	P_{l}	$D_{\max}$	D_{avg}	σ_D	d^{nq}
CSDPS	0.111	0	86.41	8.53	10.89	8.00	0.444	0	40.03	3.91	5.10	2.00
WPS	0.111	0	89.84	8.58	11.03	8.00	0.444	0	43.88	4.01	5.38	2.00
IWFQ	0.111	0	81.79	9.46	10.31	13.26	0.444	0	40.16	3.79	4.96	2.32
SBFA	0.111	0	83.25	9.52	10.69	13.36	0.443	0	42.82	3.70	4.91	2.36
CIF-Q	0.111	0	81.57	6.90	9.99	8.00	0.444	0	41.47	3.88	5.00	3.00
WFS	0.111	0	82.90	7.18	10.53	9.33	0.444	0	42.45	4.15	5.21	3.53

b. Now, we run the WFS simulation again, changing 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. There were no losses observed during the simulation run. 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. WFS can schedule low rate, low delay flows, as well as high rate, high delay flows, due to delay-bandwidth decoupling.

Example 2. Error-sensitive versus delay-sensitive flows

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 specified number of times and encounters a channel error on all its attempts. For all algorithms, we implemented the slot queue/packet decoupling as described in [6].

We consider three sources, where sources 1 and 2 are Markov-modulated Poisson processes (MMPPs), with an ON rate of 1.5 (average rate of 0.15), and source 3 is a constant source with a rate of 0.25 (i.e. packet inter-arrival time of 4). The channels for sources 1 and 2 evolve according to a two-state discrete Markov chain having a steady state probability $P_G = 0.7$ with $p_g + p_e = 0.1$. Source 3 has an error-free channel. The rate weights for all sources are $\overline{r}_i = 0.333$. Flow 1 has a retransmission bound of 8, and flow 2 has a delay bound of 100. Flow 3 has a delay bound of 100.

Table 2
Parameters and results for example 1(b): WFS.

	Sorce	$\overline{r}_i$	Φ_i	W	$D_{\max}$	D_{avg}	σ_D	d^{nq}
I	1	0.11	0.9	0.11	37.5	1.0	2.7	16
	2	0.44	0.09	0.44	40.8	2.9	4.4	23
	3	0.44	0.009	0.44	64.7	6.8	7.4	30
II	1	0.11	0.9	0.11	5.4	0.8	1.4	4
	2	0.44	0.09	0.44	11.8	2.6	2.9	5
	3	0.44	0.009	0.44	20.1	6.6	5.1	7

Table 3
Example 2: throughput (W) for three flows.

	CSDPS	WPS	IWFQ	SBFA	CIF-Q	WFS
W_1	0.2343	0.2339	0.2349	0.3091	0.2342	0.2353
W_2	0.2326	0.2341	0.2345	0.2861	0.2317	0.2342
W_3	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500

Table 3 presents the throughput results, and table 4 presents the delay results for flows 1, 2 and 3 for all the algorithms.

Flows 1 and 2 get equal throughput in all algorithms. CSDPS performs as well as WPS in this example because the error patterns for flows 1 and 2 are identical. All algorithms, except IWFQ and SBFA, have identical packet loss rates for flow 2. Flow 3 gets its due rate even though the other flows are in error. Thus, error-free flows achieve their long term throughput guarantees under all algorithms.

In IWFQ, the loss rates for flow 2 and packet delays for flow 1 are considerably less than that of the other algorithms, since IWFQ retains precedence of tags, giving priority always to a lagging flow. This results in a very high delay for the error-free flow 3, which is affected by this compensation. For SBFA, when a slot is in error an alternate flow is chosen, and its head of line packet is transmitted without checking for that flow's channel status, whereas it could have been given to another flow with a clean channel at the same time instant. The implications of this are twofold. First, if there is no other backlogged flow with a clean channel when a designated flow encounters channel error, the designated flow is still compensated, leading to a high throughput for flows 1 and 2. Secondly, when the alternate flow is in error too, the slot ends up being wasted, and since the original flow is also charged, the compensation slot is queued behind the other compensation slots. This results in a high delay and packet loss rates for the flows with channel error, as is evident here. This effect is also visible in the low delays for the error-free flow 3.

Example 3. Graceful service degradation

In this example, we look at the service degradation of leading flows. There are three flows. 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. For WFS, we bound the $E_{\max}$ and $G_{\max}$ of each flow to 50. The value of α in CIF-Q is set to 0.8. The following figures present the plot of the number of packets served over time for the various algorithms (drawn using Gnu Plot).

CSDPS. The service curves for CSDPS are shown in figure 2. Since CSDPS gives the error-prone slots of flow 1 to flows 2 and 3 uniformly until $t = 100$, both flow 2 and flow 3

Table 4
Example 2: loss rates and delay for flows 1–3.

Algorithm	Flow 1					Flow 2	Flow 3				
	P_l	$D_{\max}$	D_{avg}	σ_D	d^{nq}	P_l	P_l	$D_{\max}$	D_{avg}	σ_D	d^{nq}
CSDPS	0	175.92	21.24	25.70	102.96	0.0069	0	8.54	0.54	1.06	7.38
WPS	0	174.46	20.20	23.98	88.42	0.005	0	14.54	1.71	1.53	14.46
IWFQ	0	137.90	16.32	18.38	100.23	0.0026	0	59.13	3.22	6.00	57.39
SBFA	0.0375	4105.67	2012.82	1181.70	29.10	0.1298	0	3.95	0.29	0.70	3.95
CIF-Q	0	183.09	22.34	27.49	95.91	0.008	0	16.09	0.15	0.76	15.81
WFS	0	169.21	18.44	22.32	89.58	0.0045	0	22.16	1.90	2.62	21.79

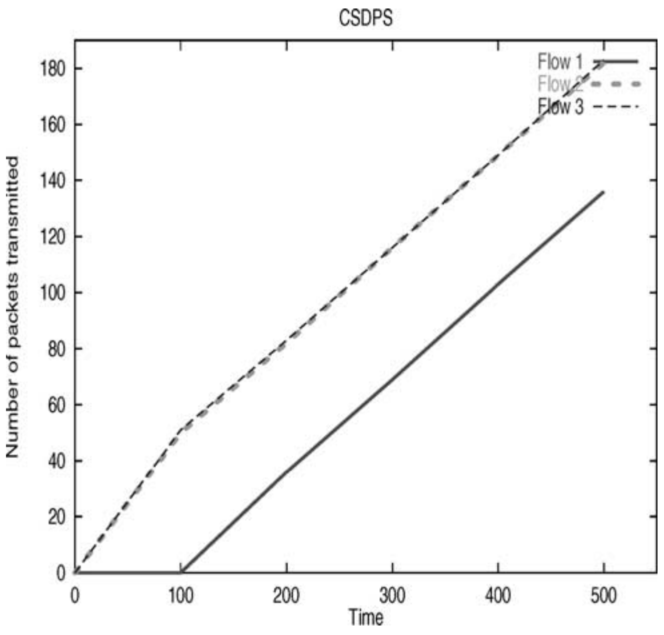

Figure 2. Service degradation plots from example 3. CSDPS.

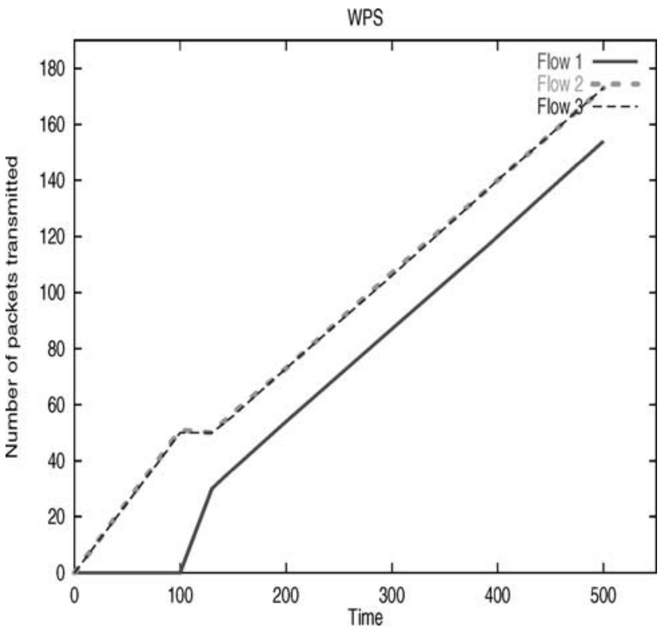

Figure 4. Service degradation plots from example 3. WPS (credit bound = 30).

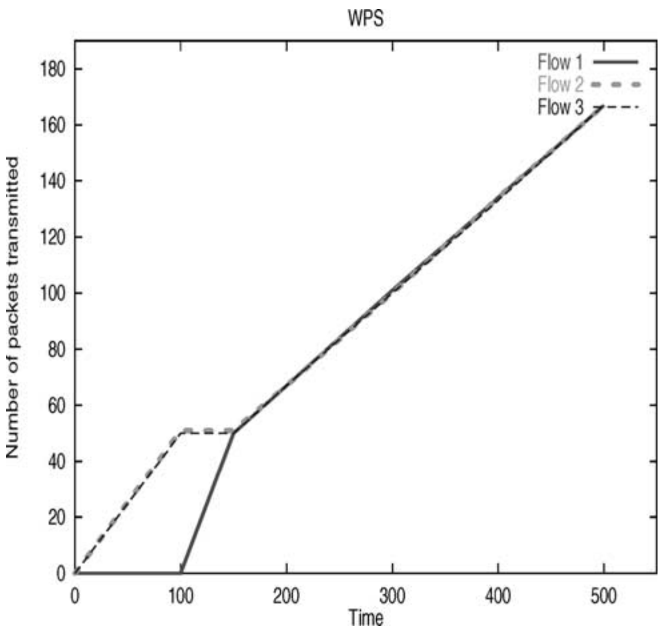

Figure 3. Service degradation plots from example 3. WPS (credit bound = 50).

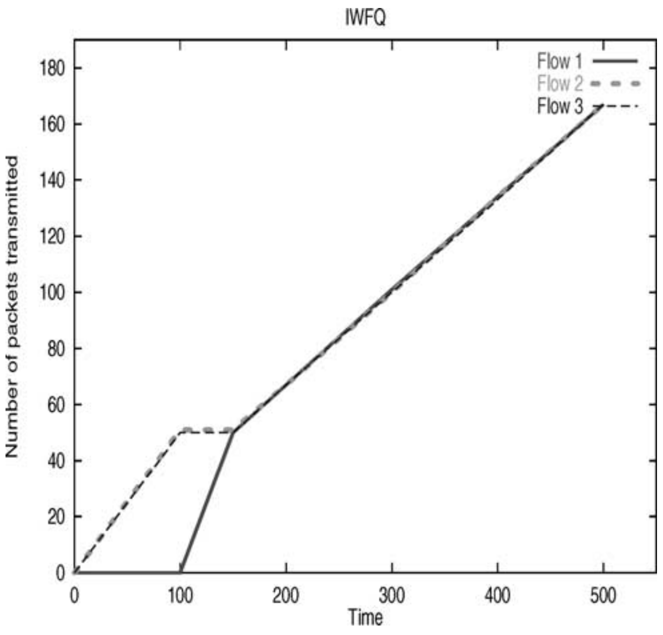

Figure 5. Service degradation plots from example 3. IWFQ.

see an increase in service. Since CSDPS does not have any mechanism for compensation, flow 3 does not receive its lost service back after $t = 100$.

WPS. WPS keeps track of the lead and lag up to the credit bound and tries to do frame swapping to compensate for the lost slots. If we bound the lead to a maximum of 50 slots for each flow, it is clear from the service curve shown in figure 3 that flow 1 captures the channel until it has given up all its lag. From figure 4, we see that if we bound the lead to 30 slots for each flow, then flow 1 loses some service since WPS does not keep track of the lag or lead beyond the bound.

IWFQ. Since IWFQ maintains precedence of tags, the lagging flow always has the minimum tag. This ensures that when the channel for a lagging error-prone flow become clean, the lagging flow captures the channel till it gives up all its lag. This is the exact behavior we observe in figure 5.

SBFA. In SBFA, the excess service is given to another flow which is then charged if the transmission is successful. A compensation slot is created corresponding to flow 1, which is in error. Since, flows 2 and 3 have clean channels at all times, they receive the excess service proportionately. At time $t = 100$, all flows including the compensation flow

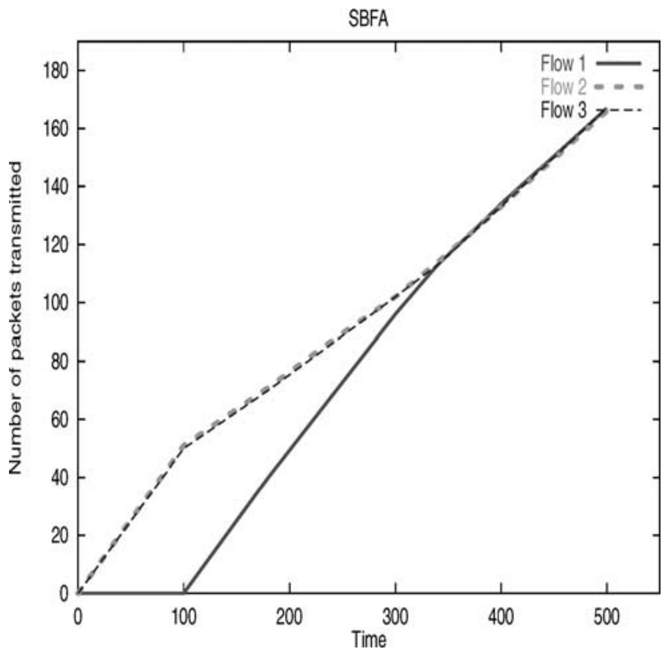

Figure 6. Service degradation plots from example 3. SBFA.

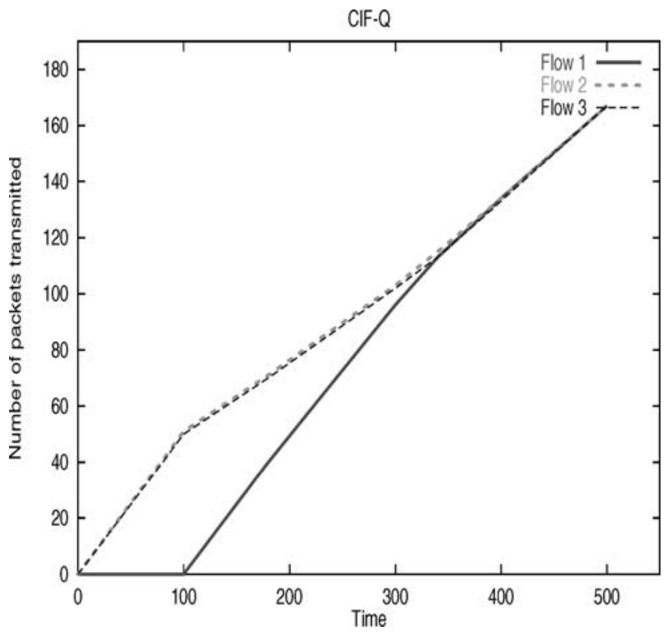

Figure 7. Service degradation plots from example 3. CIF-Q.

have equal tags. Packets of flow 1 get their normal allocation as well as the compensation allocation. Hence, they receive twice the service as flows 2 and 3, as figure 6 shows. This pattern continues till flow 1 eventually makes up for its lost service. Thus, the degradation observed for the leading flows in this case is linear.

CIF-Q. Referring to figure 7, we can conclude that CIF-Q has a linear degradation of service, the slope of which can be varied by changing the system parameter α. CIF-Q tries to distribute the excess service among all sessions. Thus, both flow 2 and flow 3 receive the same amount of excess service

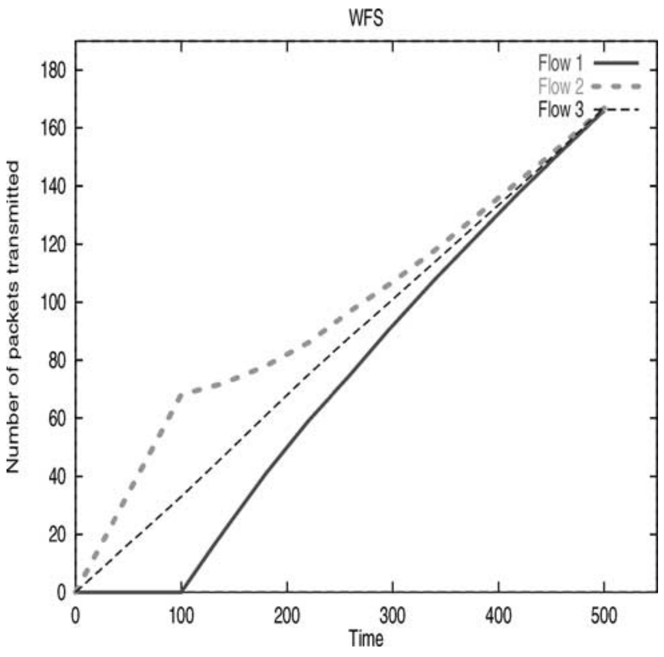

Figure 8. Service degradation plots from example 3. WFS.

which they give back to flow 1, after it has become error-free.

WFS. WFS has an exponential reduction in the degradation of service, made possible by its compensation mechanism. This is more graceful than the linear degradation observed in CIF-Q, but takes longer to compensate lagging flows. Also, WFS tries not to disturb in-sync flows, unlike CIF-Q, which distributes excess service among all clean flows. This is clearly evident from figure 8.

Graceful service degradation is important in providing short-term fairness and throughput guarantees for flows. When the degradation is abrupt, as we see in IWFQ and WPS, then leading flows do not get any short-term throughput until the lagging flows gain their lost service. If there is no service degradation however, then the algorithm fails to provide any fairness guarantees, as in CSDPS. Excess service has to be given up, and in a graceful way, such that leading flows receive some service during the compensation period. This ensures that throughput guarantees and fairness guarantees can be provided over short time windows as well. From this example, we see that only SBFA, CIF-Q and WFS ensure graceful service degradation for leading flows.

Example 4. Channel prediction

This example demonstrates the importance of channel prediction for the efficient operation of a wireless scheduling algorithm. For this purpose, let us revisit example 2. As said before, the success of one-step prediction depends on the fact that channel errors are highly correlated between slots. Now let us see what happens if this is not true. Consider the same source model as in example 2. Let $p_g + p_e = 1$ and $P_G = 0.7$. The example is now the same as in example 2 except that the channel errors now are uncorrelated between slots.

Table 5
Packet delays for flow 1 with $p_g + p_e = 1$.

Algorithm	Flow 1				Flow 2	
	$p_g + p_e = 1$				0.1	1
	D_{max}	D_{avg}	σ_D	d^{nq}	P_l	P_l
CSDPS	369.76	81.43	76.90	19.64	0.0069	0.0383
WPS	400.70	92.24	82.22	19.38	0.005	0.044
IWFQ	254.19	41.01	43.16	19.94	0.0026	0.0174
SBFA	212.00	27.78	33.47	25.50	0.016	0.009
CIF-Q	395.21	99.29	104.31	18.46	0.008	0.049
WFS	265.14	45.15	46.45	21.32	0.0045	0.0175

Table 6
Results for flow 3 in example 6.

Algorithm	W	P_l	D_{max}	D_{avg}	σ_D	d^{nq}
CSDPS	0.1666	0	120.68	9.30	14.38	99.24
WPS	0.1667	0	119.00	9.09	13.74	99.60
IWFQ	0.1666	0	119.86	9.18	13.11	98.43
SBFA	0.1667	0	139.44	16.04	23.06	95.78
CIF-Q	0.1667	0	115.93	9.56	15.25	99.00
WFS	0.1668	0	122.00	8.99	13.58	99.22

Table 5 gives the packet delays of flow 1 with $p_g + p_e = 1$, and the packet loss ratio of flow 2, with $p_g + p_e = 1$ and $p_g + p_e = 0.1$. Channel error increases by at least 300%. In SBFA, the alternate flow transmits irrespective of its channel state, and hence SBFA performs well even if channel prediction is poor. On the whole, it can be seen clearly that the worse the channel prediction, the worser the performance.

Example 5. Identical behavior

In this example, we illustrate a situation wherein all the wireless fair queueing algorithms discussed here behave in a similar way. We consider six sources, all having identical error patterns, modeled as a Markov Chain, with $p_g + p_e = 0.01$ and $P_G = 0.7$. All the sources are MMPP sources with an average rate of 0.04, and with a delay bound of 150. The characterization here is of a moderately loaded network having moderate error patterns, with a large number of sources. The simulation results for a single flow for the different algorithms is given in table 6.

The service obtained is approximately equal for all the algorithms. The delays are similar except for SBFA for the same reasons as stated in example 2. The reason for the similar performance for the algorithms is that as the number of flows increases, all flows have i.i.d. error patterns, and the offered traffic is stable but moderately heavy, so the compensation algorithms start to work approximately the same.

Example 6. Adaptive sources

A delay-sensitive flow that deletes its packets when they exceed their delay bound (due to channel error) will cease to be backlogged and thus lose its compensation. A flow can react to this packet loss by generating packets equal to the number of packets lost, at a higher rate.

Table 7
Effect of adaptive nature of source on throughput W.

D_{max}	Non-adaptive	Adaptation window				
		100	50	40	30	10
∞	0.332					
100	0.327	0.328	0.329	0.330	0.331	0.332
50	0.308	0.326	0.327	0.328	0.329	0.330

In this example, we look at the effect of the latency of adaptation on the throughput for a flow in the presence of channel errors. We have incorporated a time-window in our simulations for a flow, that determines how soon a flow reacts to this packet loss. A time-window of 20 implies that a when a source generates excess packets in reaction to a packet loss, it will be 20 time units after the loss is observed. Ideally, this time-window should be 0.

In this example, we analyze this effect through simulations of WFS. In particular, we have tried to show that the faster a flow adapts to packet loss due to delay violations, the lesser decrease in throughput is observed. Let us consider three flows. Flow 1 has an error-free channel at all times. The channel model for flow 2 evolves according to a two-state Markov chain with $p_g = 0.07$ and $p_e = 0.03$, and for flow 3 with $p_g = p_e = 0.05$. All flows are MMPP sources with $\lambda_i = 1.2$. All the flows are delay-sensitive with the delay bound = 100.

Table 7 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 with smaller time-windows, i.e. when flow 3 becomes more adaptive with respect to the rate. We see a 4% increase in throughput compared to the case when flow 3 is non-adaptive, when the delay bound is 100 for flow 3. If we reduce the delay bounds further (implying a greater number of losses), we see up to 10% increase in throughput.

6. Analytical results

In this section, we compare the analytical performance bounds of the various algorithms we described in previous sections.

6.1. Notations

We adopt the following notations for the performance comparisons described in this section: r_i is the normalized rate weight for flow i, Φ_i is the normalized delay weight for flow i in WFS, w_i is the weight (in terms of bits) for flow i in WRR, B is the maximum aggregate lag for all flows in IWFQ, α is the system parameter in CIF-Q to specify the minimum fraction of service that a leading flow retains during compensation, $F_g(t)$ is the set of lagging flows at t, $F_l(t)$ is the set of leading flows at t, F is the set of all n flows, C is the server rate, and L_p is the packet size. $C_i(t)$ is the credit/debit (in bits) of flow i at time t, where $C_i(t) > 0$ if a flow is leading; $C_i(t) < 0$ if lagging; $C_i(t) = 0$ if in-sync.

For a flow i, $W_i(t_1, t_2)$ denotes its aggregate service in bits during time interval $[t_1, t_2]$. The throughput bound for a continually backlogged flow i during $[t_1, t_2]$ is defined in terms of

Table 8
Performance of the error-free service models.

Algorithm	EF model	Throughput $W_i(t_1,t_2)$	Delay d_i^k	Fairness index $f_I(t_1,t_2)$
CSDPS	WRR	$w_i\left\lfloor \frac{C(t_2-t_1)}{\sum_{j\in F} w_j}\right\rfloor$	$\frac{L_\mathrm{p}(1+\sum_{j\in F}^{j\neq i} w_j)}{C}$	$1+\frac{L_\mathrm{p}}{w_i}+\frac{L_\mathrm{p}}{w_j}$
IWFQ	WFQ	$r_iC(t_2-t_1)-L_\mathrm{p}$	$\frac{L_\mathrm{p}}{r_iC}+\frac{L_\mathrm{p}}{C}$	$\frac{L_\mathrm{p}}{r_i}+\frac{L_\mathrm{p}}{r_j}$
WPS	WRR-S	$\frac{w_i}{\sum_{j\in F} w_j}C(t_2-t_1)-L_\mathrm{p}$	$\frac{\sum_{j\in F} w_j}{w_i}\frac{L_\mathrm{p}}{C}+\frac{L_\mathrm{p}}{C}$	$\left(\sum_{k\in F} w_k\right)\left(\frac{L_\mathrm{p}}{w_i}+\frac{L_\mathrm{p}}{w_j}\right)$
CIF-Q	STFQ	$r_iC(t_2-t_1)-r_i\sum_{i\in F} L_\mathrm{p}-L_\mathrm{p}$	$\frac{L_\mathrm{p}}{r_iC}+\frac{\sum_{j\in F} L_\mathrm{p}}{C}$	$\frac{L_\mathrm{p}}{r_i}+\frac{L_\mathrm{p}}{r_j}$
WFS	WFS-EF	$r_iC(t_2-t_1)-L_\mathrm{p}\max\left(1,\frac{r_i}{\Phi_i}\right)$	$\frac{L_\mathrm{p}}{\Phi_iC}+\frac{L_\mathrm{p}}{C}$	$\max\left(\frac{L_\mathrm{p}}{r_i},\frac{L_\mathrm{p}}{\Phi_i}\right)+\max\left(\frac{L_\mathrm{p}}{r_j},\frac{L_\mathrm{p}}{\Phi_j}\right)$
SBFA	WFQ	$r_iC(t_2-t_1)-L_\mathrm{p}$	$\frac{L_\mathrm{p}}{r_iC}+\frac{L_\mathrm{p}}{C}$	$\frac{L_\mathrm{p}}{r_i}+\frac{L_\mathrm{p}}{r_j}$

$W_i(t_1,t_2)$. The short term fairness index for two continually backlogged flows i and j during $[t_1,t_2]$ is defined to be

$$f_\mathrm{I}(t_1,t_2)=\left|\frac{W_i(t_1,t_2)}{r_i}-\frac{W_j(t_1,t_2)}{r_j}\right|.$$

In the case of WRR, it is defined to be

$$f_\mathrm{I}(t_1,t_2)=\left|\frac{W_i(t_1,t_2)}{w_i}-\frac{W_j(t_1,t_2)}{w_j}\right|.$$

The delay experienced by the kth packet p_i^k of flow i, denoted by d_i^k, is defined as the difference between its departure time $DPT(p_i^k)$ and its expected arrival time $EAT(p_i^k)$ [4]. That is, $d_i^k = DPT(p_i^k) - EAT(p_i^k)$. The expected arrival time $EAT(p_i^k)$ of packet p_i^k, which arrives at real time $A(p_i^k)$, is formally defined as

$$EAT(p_i^k)=\max\left\{A(p_i^k),\ EAT(p_i^{k-1})+\frac{L_\mathrm{p}}{r_iC}\right\},\qquad k\geqslant 1.$$

6.2. Error-free service model

The performance of the error-free service model is shown in table 8, from which we can draw the following conclusions in terms of throughput, maximum packet delay and short term fairness:

Throughput. As we can see from table 8, WFQ, WRR with spreading (WRR-S) and the error-free service model of WFS (WFS-EF) can achieve the best analytically derivable throughput bound. WRR and STFQ have coarser analytical throughput bound. Furthermore, we can see that WFS-EF achieves delay and throughput decoupling in the sense that its throughput is mainly determined by its rate weight r_i, not its delay weight Φ_i.

Packet delay. From table 8, WFQ, WRR-S and WFS-EF have the tightest analytical delay bound. WRR and STFQ have courser packet delay bound. Besides, we can see that the delay bound of WFS-EF is determined by the delay weight parameter Φ_i, not the rate parameter r_i, hence, it achieves delay and throughput decoupling.

Table 9
Performance of error-free flows in the presence of errors in other flows.

	Throughput	$d_i^k - d_i^{k,\mathrm{EF}}$
CSDPS	$W_i(t_1,t_2)\geqslant W_i^{\mathrm{EF}}(t_1,t_2)$	0
IWFQ	$W_i(t_1,t_2+T_i^{\mathrm{g}})\geqslant W^{\mathrm{EF}}(t_1,t_2)+\Delta C_i^{\mathrm{IWFQ}}(t_1,t_2)$	$T_\mathrm{d}^{\mathrm{IWFQ}}$
CIF-Q	$W_i(t_1,t_2)\geqslant W_i^{\mathrm{EF}}(t_1,t_2)+\Delta C_i^{\mathrm{CIF}}(t_1,t_2)$	$T_\mathrm{d}^{\mathrm{CIF}}$
WFS	$W_i(t_1,t_2)\geqslant W_i^{\mathrm{EF}}(t_1,t_2)+\Delta C_i^{\mathrm{WFS}}(t_1,t_2)$	$T_\mathrm{d}^{\mathrm{WFS}}$
SBFA	$W_i(t_1,t_2)\geqslant W_i^{\mathrm{EF}}(t_1,t_2)$	0

Short term fairness. WRR achieves the worst short term fairness, and other algorithms achieve comparable short term fairness index.

6.3. Error-free flows in the presence of channel errors

In this subsection, we compare the performance of error-free flows in the presence of channel errors. A detailed characterization of throughput and packet delay bounds is shown in table 9, where $d_i^{k,\mathrm{EF}}$ denotes the packet delay for kth packet of flow i in the error-free service model.

6.3.1. Throughput

We proceed with a generic throughput bound which may characterize both short-term and long-term throughput behavior of these scheduling algorithms, and then describe their properties (see table 9).

Theorem 1 (Throughput bounds). Consider a continually backlogged flow i during interval $[t_1,t_2]$. Let $W_i(t_1,t_2)$ be the service received by error-free flow i during $[t_1,t_2]$, then

$$W_i(t_1,t_2)\geqslant W_i^{\mathrm{EF}}(t_1,t_2)+C_i(t_2)-C_i(t_1),\qquad(2)$$

where $W_i^{\mathrm{EF}}(t_1,t_2)$ denoted the service that flow i has received in its error-free service during $[t_1,t_2]$.

Property 1 (CSDPS). The CSDPS algorithm does not have any notion of lead and lag, i.e. $C_i(t)\equiv 0\ \forall t$. Therefore, as long as flows are continually backlogged, leading flows do not give up services and lagging flows do not receive compensation.

Property 2 (IWFQ). For a continually backlogged error-free flow i over $[t_1, t_2]$,

- if flow i is leading, then we have
$$T_i^{\mathrm{g}} = \frac{C_i(t_1)\sum_{j\in F_{\mathrm{g}}} r_j}{Cr_i} + \frac{\sum_{j\in F_{\mathrm{g}}} |C_j(t_1)|}{C},$$
and
$$\Delta C_i^{\mathrm{IWFQ}}(t_1, t_2) \geqslant -C_i(t_1);$$
- if flow i is lagging, then we have
$$T_i^{\mathrm{g}} = \frac{\sum_{j\in F_{\mathrm{g}}} |C_j(t_1)|}{C}$$
and
$$\Delta C_i^{\mathrm{IWFQ}}(t_1, t_2) = 0.$$

Property 3 (CIF-Q). Consider the case $C_i(t_1) = C_0$. For a continually backlogged error-free flow i over $[t_1, t_2]$,

- if flow i is leading, then the following holds:
$$C_i(t_2) - C_i(t_1) \geqslant \Delta C_i^{\mathrm{CIF}}(t_1, t_2),$$
where
$$\Delta C_i^{\mathrm{CIF}}(t_1, t_2) := \min\bigl(0, -(1-\alpha)r_i C(t_2 - t_1) + L_{\mathrm{p}} + (1-\alpha)(nr_i + 1)L_{\mathrm{p}}\bigr),$$
- if flow i is lagging, then the following holds:
$$C_i(t_2) - C_i(t_1) \geqslant \Delta C_i^{\mathrm{CIF}}(t_1, t_2),$$
where
$$\Delta C_i^{\mathrm{CIF}}(t_1, t_2) := r_i \sum_{j\in F_1} \bigl|\Delta C_j^{\mathrm{CIF}}(t_1, t_2)\bigr| - (n-1)r_i L_{\mathrm{p}} - L_{\mathrm{p}}.$$

Property 4 (WFS). Consider the case $C_i(t_1) = C_0$. For a continually backlogged error-free flow i over $[t_1, t_2]$,

- if flow i is leading, then its credit is updated as
$$C_i(t_2) - C_i(t_1) \geqslant \Delta C_i^{\mathrm{WFS}}(t_2, t_1),$$
where
$$\Delta C_i^{\mathrm{WFS}}(t_2, t_1) := C_0\bigl(1 - \mathrm{e}^{-r_i C(t_2 - t_1)/C_i^{\max}}\bigr);$$
- if flow i is lagging, then its credit is updated as
$$\begin{aligned} &C_i(t_2) - C_i(t_1) \\ &\geqslant \Delta C_i^{\mathrm{WFS}}(t_2, t_1) \\ &:= \min\Biggl\{|C_i(t_1)|, \sum_{k\in F_1(t_1)} \frac{C_k(t_1) r_k C(t_2 - t_1)}{r_k C(t_2 - t_1) + C_i^{\max}} \times \frac{|C_i(t_1)|}{\sum_{j\in F_{\mathrm{g}}(t_1)} |C_j(t_1)|}\Biggr\};\end{aligned}$$
- if flow i is in-sync, then its credit is updated as
$$C_i(t_2) - C_i(t_1) = \Delta C_i^{\mathrm{WFS}}(t_2, t_1) := 0.$$

Remark 1. In CIF-Q, the compensation is distributed among lagging flows according to their weight r_i; in WFS, the compensation is distributed among lagging flows according to a WRR, where the weight is a flow's lag. Besides, in both CIF-Q and IWFQ, in-sync flows may be disturbed, but in WFS in-sync flows are not disturbed.

Remark 2 (SBFA and CSDPS). It might be misleading from table 9 that SBFA and CSDPS seem to perform the best. This is not true since in CSDPS, lagging flows will never receive compensation; in SBFA, there is a fundamental conflict for fairness between the service allocated to a flow and the prereserved fraction, therefore, it is not exactly fair queueing in the sense that its error-free service and fairness are defined by excluding the prereserved fraction.

6.3.2. Packet delay

In table 9, we provide the packet delay for an error-free lagging flow, since the cases for leading and in-sync flows are straightforward.

Property 5. For a packet of error-free lagging flow i, the following is true for its packet delay d_i^k (see table 9):

- in IWFQ,
$$T_{\mathrm{d}}^{\mathrm{IWFQ}} = \frac{|C_i(t)|}{r_i C} + \frac{B}{C};$$
- in CIF-Q,
$$T_{\mathrm{d}}^{\mathrm{CIF}} = \frac{|C_i(t)|}{r_i C} + \min\biggl\{\frac{L_{\mathrm{p}}}{r_i C}, T_{\mathrm{m}}^{\mathrm{CIF}}\biggr\},$$
where
$$T_{\mathrm{m}}^{\mathrm{CIF}} = \frac{L_{\mathrm{p}}}{(1-\alpha) r_i r_{\min} C} + \biggl(\frac{1/r_i + n - 1 + \alpha}{r_{\min}(1-\alpha)} + n + \frac{1}{r_{\min}}\biggr)\frac{L_{\mathrm{p}}}{C}$$
with $r_{\min} = \min_{j\in F_1} r_j$;
- in WFS,
$$T_{\mathrm{d}}^{\mathrm{WFS}} = \frac{|C_i(t)|}{Cr_i} + \min\biggl\{\frac{L_{\mathrm{p}}}{r_i C}, T_{\mathrm{m}}^{\mathrm{WFS}}\biggr\},$$
where $T_{\mathrm{m}}^{\mathrm{WFS}}$ is calculated from equation
$$CT_{\mathrm{m}} \sum_{k\in F_1(t)} \frac{C_k(t) r_k / L_{\mathrm{p}}}{r_k C T_{\mathrm{m}} + C_i^{\max}} = \frac{\sum_{j\in F_{\mathrm{g}}(t)} |C_j(t)|}{|C_i(t)|}.$$

6.4. Service degradation of leading flows

In this section, we compare the service degradation of leading flows in IWFQ, CIF-Q and WFS.

Theorem 2 (Graceful service degradation for CIF-Q and WFS). Consider a leading backlogged flow i over a time interval $[t_1, t_2]$. Assume $C_i(t_1) = C_0$ at time t_1. Then, for any time $t \in [t_1, t_2]$:

1. In WFS, its credit $C_i(t)$ and instantaneous rate $r_i(t)$ are given by
$$C_i(t) \geqslant C_0\,\mathrm{e}^{-r_i C(t-t_1)/C_i^{\max}}$$
and
$$r_i(t) \geqslant r_i C\left(1 - \frac{C_0}{C_i^{\max}}\,\mathrm{e}^{-r_i C(t-t_1)/C_i^{\max}}\right).$$
2. In CIF-Q, its credit $C_i(t) \geqslant 0$ is given by
$$C_i(t) \geqslant C_0 - \max\bigl(0, (1-\alpha)r_i C(t-t_1) - L_\mathrm{p} - (1-\alpha)(1+nr_i)L_\mathrm{p}\bigr),$$
and its instantaneous rate $r_i(t)$ is given by
$$r_i(t) \geqslant r_i C(1-\alpha).$$

Theorem 3 (Service starvation time for IWFQ). For a leading flow with lead C_i at time t, the maximum service starvation time T_i^{sv}, defined as the maximum time that a leading flow does not receive any service compared to its error-free service, is given by
$$T_i^{\mathrm{sv}} = \frac{C_i\bigl(\sum_{j\in F_\mathrm{g}} r_j\bigr)}{Cr_i} + \frac{\sum_{j\in F_\mathrm{g}} |C_j(t)|}{C}.$$

Remark 3 (SBFA and CSDPS). In SBFA, though leading flows do not give up services directly, the server has to preallocate a fraction of bandwidth for compensation, thus effectively reducing the throughput for leading flows. In CSDPS, the lagging flows do not receive any compensation due to channel error.

6.5. Service capture effect by lagging flows

The service capture effect happens in one of the two scenarios: (1) the entire channel is captured by a subset of lagging flows, and other flows are starved out of service for certain period of time; and (2) the compensation service is captured by a subset of lagging flows, and other lagging flows are starved out of compensation for certain period of time. As we can see from the following theorems, all algorithms except WFS suffer from one of the two capture effects. Note that it does not apply for CSDPS since no compensation effort is made there.

Property 6 (Channel capture in IWFQ). In a worst case scenario, the maximum channel capture time $T_{i,\max}^{\mathrm{cap}}$ by a lagging flow i, which has a specified maximum lag B_i, is given by $T_{i,\max}^{\mathrm{cap}} = B_i/C$.

Property 7 (Compensation capture in CIF-Q). In CIF-Q, the difference between the normalized compensation (in virtual time) of any two lagging error-free flows i and j is $-L_\mathrm{p}/r_j \leqslant c_i - c_j \leqslant L_\mathrm{p}/r_i$. Therefore, a lagging flow with a large lag but with small weight may be starved out compensation for certain period of time.

Property 8 (Compensation capture in SBFA). In SBFA, since the tagging history of lagging flows is maintained by the Long Term Fairness Server (LTFS) in the same way as in IWFQ, it suffers from the same capture effect as IWFQ for compensation service. That is, in a worst case scenario, the maximum capture time for compensation service by a lagging flow i, which has a lag b_i, is given by $T_{i,\max}^{\mathrm{cap}} = b_i/B_{\mathrm{resv}}$, where B_{resv} is the preallocated capacity for compensation.

7. Summary

Wireless fair queueing is an important emerging area of wireless network research because simple best-effort scheduling of flows is inadequate in scarce and heavily loaded channels. While several wireless fair queueing algorithms have been proposed in literature, to our knowledge, this is the first work that proposes a unifying architecture and a detailed performance evaluation of different wireless fair queueing algorithms.

We have presented the wireless fair service, which captures the key requirements of wireless scheduling algorithms. We have presented a unified wireless fair queueing architecture, and mapped seven of the candidate wireless fair queueing algorithms onto this architecture. A detailed simulation and analysis based performance evaluation of these algorithms shows that CIF-Q and WFS satisfy all the properties of wireless fair service.

References

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

[2] P. Bhagwat, P. Bhattacharya, A. Krishma and S. Tripathi, Enhancing throughput over wireless LANs using channel state dependent packet scheduling, in: *IEEE INFOCOM'96* (April 1996).

[3] A. Demers, S. Keshav and S. Shenker, Analysis and simulation of a fair queueing algorithm, in: *ACM SIGCOMM'89* (August 1989).

[4] P. Goyal, H.M. Vin and H. Chen, Start-time Fair Queueing: A scheduling algorithm for integrated service access, in: *ACM SIGCOMM'96* (August 1996).

[5] S.J. Golestani, A self-clocked fair queueing scheme for broadband applications, in: *IEEE INFOCOM'94* (April 1994).

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

[7] M. Srivastava, C. Fragouli and V. Sivaraman, Controlled multimedia wireless link sharing via enhanced class-based queueing with channel-state-dependent packet scheduling, in: *IEEE INFOCOM'98* (March 1998).

[8] T.S. Ng, I. Stoica and H. Zhang, Packet fair queueing algorithms for wireless networks with location-dependent errors, in: *IEEE INFOCOM'98* (March 1998).

[9] P. Ramanathan and P. Agrawal, Adapting packet fair queueing algorithms to wireless networks, in: *ACM MOBICOM'98* (October 1998).

[10] S. Lu, T. Nandagopal and V. Bharghavan, Fair scheduling in wireless packet networks, in: *ACM MOBICOM'98* (October 1997).

[11] D. Eckhardt and P. Steenkiste, Improving wireless LAN performance via adaptive local error control, in: *IEEE ICNP'98*, Austin (October 1998).

[12] E.O. Elliot, Estimates of error rates for codes on burst-noise channels, Bell Systems Technical Journal 42 (September 1963) 1977–1997.

[13] 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).

[14] S. Floyd and V. Jacobson, Link-sharing and resource management models for packet networks, IEEE/ACM Transactions on Networking 3(4) (August 1995) 365–386.

Thyagarajan Nandagopal is a PhD candidate in the Electrical and Computer Engineering Department at the University of Illinois and is affiliated with the TIMELY Research Group. His research focuses on providing Quality-of-Service in wireless networks.
E-mail: thyagu@timely.crhc.uiuc.edu

Songwu Lu is an Assistant Professor in the Computer Science Department at the University of California at Los Angeles. His research focuses on developing wireless scheduling and medium access protocols that provide Quality-of-Service.
E-mail: slu@cs.ucla.edu

Vaduvur Bharghavan is an Associate Professor in the Electrical and Computer Engineering Department at the University of Illinois, where he heads the TIMELY Research Group. His research interests are in mobile computing and computer networking.
E-mail: bharghav@timely.crhc.uiuc.edu