Design and Analysis of an Algorithm for Fair Service in Error-Prone Wireless Channels

Songwu Lu, Thyagarajan Nandagopal and Vaduvur Bharghavan
Coordinated Sciences Laboratory, University of Illinois at Urbana-Champaign
E-mail: {slu,thyagu,bharghav}@timely.crhc.uiuc.edu

In order to support diverse communication-intensive real-time and non real-time data flows over a scarce, varying and shared wireless channel with location-dependent and bursty errors, we define a service model that has the following characteristics: *short-term fairness* among flows which perceive a clean channel, *long-term fairness* for flows with bounded channel error, *worst-case delay bounds* for packets, *short-term throughput bounds* for flows with clean channels and *long-term throughput bounds* for all flows with bounded channel error, *expanded schedulable region*, and support for both *delay sensitive and error sensitive* data flows. We present the wireless fair service algorithm, and show through both analysis and simulation that it achieves the requirements of the service model in typical wireless network environments. The key aspects of the algorithm are the following: (a) an enhanced fair queueing based service scheme that supports decoupling of delay and bandwidth, (b) graceful service compensation for lagging flows and graceful service degradation for leading flows, (c) support for real-time delay sensitive flows as well as non real-time error sensitive flows, and (d) an implementation within the framework of the simple and robust CSMA/CA wireless medium access protocol.

Keywords: Wireless Scheduling, Fair Queueing, Wireless Networks, Wireless Fair Service

1. Introduction

In recent years, there has been a tremendous growth in the wireless networking industry. With the increasing usage of mobile and wireless networks in both indoor and outdoor environments, the issue of providing fair channel access among multiple contending hosts over a scarce and shared wireless channel has come to the fore. In wireline networks, fair queueing has long been a popular paradigm for providing fairness and bounded delay access [1,2]. However, adapting fair queueing to wireless networks is non-trivial because of the unique problems in wireless channels: (a) the *channel capacity is dynamically varying*, (b) *channel errors are location-dependent and bursty* in nature, (c) there is *contention* in the channel among multiple mobile hosts, (d) there are hidden and exposed stations, (e) *mobile hosts do not have global channel state* (in terms of which other hosts that are contending for the same channel have packets to transmit, etc.), (f) the scheduling must take care of both *uplink and downlink flows*, and (g) mobile hosts are often constrained in terms of processing power and battery power. Thus, any wireless scheduling and channel access algorithm must work within the constraints imposed by the environment.

At the same time, in order to support communication-intensive real-time and non real-time data flows over a scarce, varying and shared channel, the service model must have the following characteristics: (a) it must provide *short-term fairness* among flows which perceive a clean channel, (b) it must provide *long-term fairness* for flows with bounded channel error, (c) it must provide *delay bounds* for packets, (d) it must provide *short-term throughput bounds* for flows with clean channels and *long-term throughput bounds* for all flows with bounded channel error, (e) it must *expand the schedulable region* by accepting flows with different decoupled delay/bandwidth requirements, and (f) it must support both *delay sensitive and error sensitive* data flows.

In this paper, we propose a service model that seeks to provide channel-conditioned deterministic guarantees with the above six properties for uplink and downlink flows in a cellular environment. We term this service model as the *'wireless fair service model'*. We also present a novel wireless scheduling and channel access algorithm that achieves the wireless fair service model and provide a simple CSMA/CA based channel access protocol for packet cellular networks that can be easily implemented in mobile hosts with low processing and power requirements. We analyze its properties, and through simulations, we show that the algorithm performs quite well, achieving the wireless fair service model for typical wireless communication environments with diverse flow requirements.

The rest of the paper is organized as follows. Section 2 discusses related work and the motivation for this paper. Section 3 presents the wireless fair service algorithm and describes a practical implementation of the scheduling algorithm and medium access protocol. Section 4 analyzes the properties of our algorithm and shows that it achieves the wireless fair service model. Section 5 evaluates the performance of our algorithm through simulation. Section 6 concludes the paper.

2. Background and Motivation

In this section, we first present the channel model. We then review related work in wireline and wireless fair queueing. Finally, we motivate our work and briefly compare our approach to related work.

2.1. Channel Model

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

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

2.2. Wireline Fair Queueing

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

2.3. Wireless Fair Queueing

Wireline fair queueing algorithms do not account for wireless channel issues such as location dependent errors and wireless medium access. In order to adapt fair queueing to the wireless domain, we proposed the Idealized Wireless Fair Queueing algorithm (IWFQ) in a related work [4]. Specifically, IWFQ allows for location-dependent and bursty channel error, and for uplink and downlink flows in a shared broadcast channel. IWFQ defines two types of service: the *error-free service*, where all flows perceive a clean channel at all times, and the *real service*, where some flows may perceive dirty channels at any given time. Each flow is labeled as being 'leading', 'lagging', or 'in sync' depending on whether its real service is ahead of, behind, or in accordance with its error-free service. At each time, the packet with the minimum finish tag on a clean channel is transmitted. Thus, flows that experience clean channels can lead their error-free service at the expense of flows that perceive channel errors. Lagging flows are allowed to retain their precedence in terms of finish tags; thus, when a lagging flow perceives a clean channel, its packets will have the highest precedence (i.e. lowest finish tags), and it will be able to catch up on its lag. We bound the amount of lag and lead. Bounding the lag is equivalent to bounding the amount of compensation that a flow can receive due to past channel errors. Bounding the lead is equivalent to bounding the amount of compensation that a flow will pay for having received additional service due to channel error in other flows. Based on the bounds of lag and lead, and the ability to maintain precedence for lagging flows, IWFQ is able to establish channel conditioned guarantees for maximum delay and minimum throughput. While IWFQ is an ideal service model, we proposed the Wireless Packet Scheduling algorithm (WPS) to address the practical issues in wireless scheduling and medium access such as having the base station schedule uplink and downlink flows, propagating information about uplink flows to the base station, medium access, etc.

In a recent work, Ng et. al. [5] proposed a new service model, called Channel-condition Independent Fair (CIF) model, that subsumes four key properties that a packet fair queueing algorithm should possess in a wireless environment: (a) delay and throughput guarantees for error-free flows, (b) short-term fairness for error-free flows, (c) long-term fairness for error-prone flows, and (d) graceful service degradation for leading flows. They also presented a CIF-Q algorithm that achieves the CIF service model. CIF-Q improves on IWFQ in two key aspects: (a) degradation of service for leading flows is graceful and (b) compensation among lagging flows is fair. However, decoupling of delay and bandwidth is not considered in CIF and CIF-Q, and practical MAC layer issues associated with wireless fair queueing are not addressed. Additionally, in sync flows are disturbed and become leading flows when a lagging flow leaves the set of active flows. In some pathological cases, the service degradation model in CIF-Q breaks down.

IWFQ has the property that a flow that has perceived channel error for a long time and then perceives a clean channel can essentially capture the channel for short time periods (till it catches up with the error-free service). Both the WPS algorithm in [4] and the CIF algorithm in [5] address this problem by ensuring that compensation is given more gradually to lagging flows. While WPS provides a weighted round-robin implementation of IWFQ, CIF maintains the fair queueing properties of IWFQ.

Ramanthan et. al. propose a generalized framework for channel dependent scheduling called the Server Based Fairness Approach [6], which is based on the assumption that all flows which perceive clean channel states must receive their promised service and not a fraction of the service as in [4,5]. Any flow that is unable to transmit due to channel error is supplemented with additional bandwidth using special sessions called Long-Term Fairness Servers (LTFS). The LTFS sessions are scheduled similar to packet flows, and cause lagging flows to receive compensation, thus ensuring long-term throughput guarantees for all flows. However, it is possible for a flow to receive compensation even when there are no other active flows to utilize the service given up by this flow. The compensation model in SBFA may cause in sync flows to be disturbed. Additionally, worst case delay bounds and fairness in SBFA are much coarser than in CIF-Q or the algorithm presented in this paper.

In addition to the algorithms described above [4–6], there have recently been a number of other wireless channel allocation algorithms that seek to provide fair service [7–9]. To our knowledge, none of these algorithms formally provide the quality of service parameters that we seek to provide with the wireless fair service model.

2.4. Service Model and Motivation

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

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

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

An inherent limitation of fluid fair queueing is that the delay observed by the packets of a flow is tightly coupled with the fraction of the channel given to the flow among all backlogged flows. IWFQ [4] and CIF-Q [5] suffer from this limitation too. Additionally, IWFQ does not provide short term fairness among flows with clean channels; CIF provides short term fairness, but does not account for any of the practical issues in wireless medium access. As described in Section 1, the wireless fair service model accounts for the above issues, and the wireless fair service algorithm presented in this paper achieves the goals of the model.

3. Wireless Fair Service Algorithm

In this section, we present the wireless fair service algorithm and wireless medium access protocol. Our algorithm is composed of five key components that work in concert in order to achieve wireless fair service:

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

In this section, we will describe each of the components of the algorithm, and finally put together the different components. In the next section, we show via theoretical analysis, that our algorithm achieves wireless fair service, while in Section 5, we will verify the same via simulation experiments.

3.1. The error-free service model

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

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

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

Conceptually, we have a two-level service model, in which a flow i is drained into the scheduler at rate r_i, but served at rate Φ_i. The virtual time moves similar to fluid fair queueing, but each packet is assigned a finish tag that is the sum of its start tag (when it is expected to arrive in virtual time) and its deadline (the maximum number of rounds, in virtual time, [1] that the packet is expected to wait before being served). The introduction of Φ_i enables us to switch the schedule among arrived packets in a way that decouples the long term rate of arriving packets (according to r_i) from the delay bounds for packets (according to Φ_i and r_i). The lookahead parameter ϱ provides a measure of the number of packets over which we can locally switch the schedule of packets without disrupting the long-term rate. Basically, a lookahead of ϱ enables the scheduler to 'look ahead in virtual time' by ϱ rounds, and swap the service order of packets that have already arrived in current time with packets that arrive in the future but have shorter deadlines. In particular, if $\varrho = 0$, then the error-free weighted fair service algorithm is basically the worst case fair weighted fair queueing algorithm (WF^2Q)[10] with deadline-based swapping among the head-of-line packets. If $\varrho = \infty$, then the error-free weighted fair service algorithm is similar to the earliest deadline first algorithm, but with backlog compensation as in fair queueing (i.e. if a flow does not have a packet to transmit at any time, it cannot later reclaim this lost slot, since virtual time moves forward). For the rest of this paper, we will assume $\varrho = \infty$.

The key property satisfied by the basic model is that it can decouple delay and bandwidth in a fair queueing scheduling framework. The reason for selecting this

[1] Virtual time moves no slower than real time, if $\sum_{i \in F} r_i = 1$.

approach rather than going directly with the earliest deadline first approach is that fair queueing can better ensure short-term fairness among flows, is better to arbitrate among non real-time as well as real-time flows, and can accommodate more sophisticated compensation models in the error case.

3.2. Slot Queues and Packet Queues

In most related work on both wireline and wireless fair queueing [1,11], packets are tagged as soon as they arrive. This works well if we assume no channel error, i.e. a scheduled packet will never be lost. However, in a wireless channel, a lost packet may need to be retransmitted for an error-sensitive flow. Retagging the packet will cause it to join the end of the flow queue and thus cause packets to be delivered out of order.

Fundamentally, there needs to be a separation between 'when to send the next packet', and 'which packet to send next'. The first question should be answered by the scheduler, while the second question is really a flow-specific decision and should be beyond the scope of the scheduler. In order to decouple the answers to these two questions, we propose a mechanism in [4], where we use one additional level of abstraction - we tag 'slots' and each slot points to a flow queue. At each time, the scheduler determines which slot will get access to the channel. The HOL packet in the corresponding flow queue is then transmitted. When a packet arrives at a flow queue, a corresponding slot is generated in the slot queue of the flow. The number of slots in the slot queue at any time is exactly the same as the number of packets in the flow queue.

Providing this additional level of abstraction is crucial to being able to achieve the wireless fair service model. Error-sensitive flows will not delete the HOL packet upon channel error during transmission, but delay-sensitive flows may delete the HOL packet once it violates its delay bound. Likewise, the flow may have priorities in its packets, and may choose to discard an already queued packet in favor of an arriving packet when its queue is full [12]. In our scheduling model, we support any queueing and packet dropping policy at the flow level because we decouple slot queues from flow queues.

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

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

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

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

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

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

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

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

Case 1 - i is leading and has designated slot s for transmitting one of its packets: If i perceives a clean channel, then it transmits its HOL packet.
Otherwise, if there is a backlogged lagging flow j that perceives a clean channel, then j is allocated the slot, $E(i)$ is decremented by 1, and $G(j)$ is decremented by 1.
Otherwise, if there is a backlogged leading flow j which perceives a clean channel and whose lead is less than its lead bound, i.e. $\exists j, E(j) < E_{\max}(j)$, then j is allocated the slot, $E(i)$ is decremented by 1, and $E(j)$ is incremented by 1.
Otherwise, if there is any backlogged in sync flow j which perceives a clean channel, then j is allocated the slot, $E(i)$ is decremented by 1, and $E(j)$ is incremented by 1.
Otherwise, if there is any backlogged flow that per-

ceives a clean channel, it is allocated the slot. If there is no backlogged flow with a clean channel, then the slot is wasted.

The sequence of operations is shown in lines 3-22 in Figure 1.

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

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

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

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

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

The sequence of operations is shown in lines 25-42 in Figure 1.

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

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

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

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

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

The sequence of operations is shown in lines 43-62 in Figure 1.

The channel allocation algorithm corresponding to these four states is detailed in Figure 1.

Essentially, our slot allocation algorithm tries to avoid wasting slots, tries to avoid disturbing in sync flows, and tries to reduce lag before increasing lead when swapping slots. At all times, the sum of all the lags and all the leads in the channel is 0.

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

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

43 else /* an in-sync or lagging flow with lag >= 0 */
44   if( slot_state(marked_flow) == CLEAN )
45     return marked_flow;  /* transmit HOL packet */
46   else
```

```
    compensated_flow =
          select_clean_lagging_flow_from_WRR();
    if(compensated_flow != NULL )
    /* swap between lagging/in-sync and
                    lagging flows (Case 3-4) */
      marked_flow->lag ++;
      compensated_flow->lag--;
  else       /* No clean lagging flow */
    compensated_flow = pick_leading_flow();
    /* find clean backlogged leading flow
                          with E(i) < E_max(i)*/
    if( compensated_flow == NULL)
      compensated_flow = pick_clean_in-sync_flow();
    if( compensated_flow!=NULL)
    /* swap btw. lagging/in-sync and
             leading/in-sync flow (Case 3-4) */
      marked_flow->lag ++;
      compensated_flow->lead ++;
  else
    compensated_flow =
          pick_any_clean_backlogged_flow();
    if (compensated_flow == NULL)
      wasted_slots ++;
return compensated_flow;
```

Figure 1. Pseudo Code for the Wireless Channel Allocation Algorithm

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

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

3.4. Compensation Model

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

3.4.1. Rate compensation from leading flows

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

Consider a leading flow i with a rate weight of r_i, a lead of $E(i)$, and a maximum lead of $E_{\max}(i)$. i hierarchically decomposes itself into two flows, i^c and i^t (see Figure 2), with rates of $r_i.E(i)/E_{\max}(i)$, and $r_i.(1 - E(i)/E_{\max}(i))$. i^c is designated to be the compensation flow, while i^t is designated to be the transmission flow. When i is allocated a slot, it hierarchically schedules it among i^c and i^t. All slots belonging to i^c are relinquished. Note that as the lead decreases, the rate for i^c decreases linearly. This has the property of graceful degradation of service for a leading flow, as shown in the analysis in Section 4.

3.4.2. Rate compensation to lagging flows

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

Our approach is to provide channel access to backlogged lagging flows from two sources: (a) the normal rate-based slot allocation, and (b) distributing the relinquished slots from leading flows fairly among the lagging flows. We maintain a 'compensation weighted round robin' slot allocation (see Figure 2) among the lagging flows, where the weight of each lagging flow i is $G(i)$. Associated with the compensation WRR is a current pointer, which identifies the next lagging flow in the WRR that should receive a compensation slot. When a leading flow relinquishes a slot, we traverse through the WRR looking for the first flow that perceives a clean

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

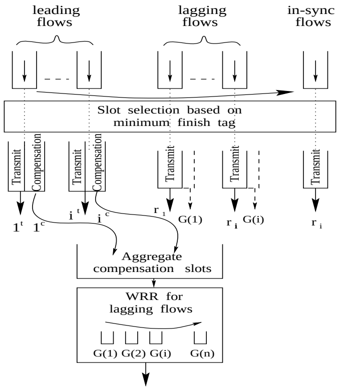

Figure 2. A Schematic for the Wireless Fair Service (WFS) Algorithm

channel, and allocate the relinquished slot to that flow. Thus, lagging flow receive slots from two sources at any time t: slots at a rate of $r_i/\sum_{j\in B(t)} r_j$ from the basic wireless fair service algorithm, and slots at a rate of $(G(i)/\sum_{k\in F} G(k)).\sum_{k\in lead}(E(k).r_k)/(E_{\max}(k).\sum_{j\in B(t)} r_j)$ from the compensation WRR. As is clear from the algorithm, the compensation model allocates the compensation slots fairly among the lagging flows.

3.5. Wireless Medium Access Protocol

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

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

3.5.1. The basic MAC Protocol Mechanism

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

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

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

Note that we have assumed that the wireless channel errors are bursty and highly correlated between successive slots. Thus, if a mobile host does not receive the base station's piggybacked transmission list for slot i in the Data or ACK packet of slot $i-1$, then the mobile host assumes that it perceived a channel error and does not contend for the next slot. Hence, only stations which are in the transmission list for slot i and perceive a clean channel in slot $i-1$ will contend for sending the RTS in slot i. Furthermore, the contention among the transmitting stations is very well controlled by the base station, though it involves very few handshakes, unlike the algorithm in WPS, or similar algorithms in fair wireless medium access such as DQRUMA [13].

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

3.5.2. Notifying the base station about backlogged flows

A flow may change its backlog status under three conditions: (a) it was previously backlogged but received a new run of packets, (b) it just finished transmission of the last packet in a run, and (c) it deleted all its backlogged packets because they violated their delay bounds.

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

Once the base station has the information about the newly backlogged flows, it performs the wireless fair service algorithm as detailed in the previous section.

3.6. Implementation Complexity

It should be noted that if the designated flow is able to transmit at each time instant, i.e., the selected flow perceives a clean channel at the time that it is being selected, then our algorithm has the same computational complexity as the WFQ algorithm. In fact, many computationally efficient approximations to Fluid Fair Queueing, such as STFQ [3] , SCFQ [11] can be directly applied to our framework.

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

4. Analysis of the Wireless Fair Service Algorithm

In this section, we analyze[3] the wireless fair service algorithm (assuming perfect MAC and perfect channel knowledge) under the following two conditions:

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

[3] The non-trivial proofs are presented in the Appendix.

4.1. Performance of the Error-Free Service Model

4.1.1. Schedulability Condition with $\varrho \in [0, \infty]$

We adopt the same notations as in [14]. Let the nonnegative vector $\vec{D} = (\bar{D}_1, \bar{D}_1, \ldots, \bar{D}_N)$ be a list of required upper bounds on delay so that no packets from flow i are delayed by $\bar{D}_i$. The vector $\vec{D}$ is *schedulable* under a scheduling policy π if for all arrival patterns that satisfy the stability condition, and for each flow $i \in F$, no packet of flow i is delayed by more than D_i (measured in seconds or bits). The *schedulable region* Ω^π of the policy π is the set of all vectors schedulable under π. We define a scheduling policy π^* to be *delay-optimal* if $\Omega^\pi \subseteq \Omega^{\pi^*}$ for all policies $\pi \in P$ where P denotes a class of admissible policies. That is, a delay-optimal policy has the *largest* schedulable region among a class of admissible scheduling policies.

The following two lemmas establish the sufficient conditions for a schedulable non-preemptive policy.

Lemma 4.1. Consider a lookahead window $\varrho \in [0, \infty]$. Let $\bar{D}_1 \le \bar{D}_2 \le \ldots \le \bar{D}_N$ where $\bar{D}_i = \frac{L_{\max}}{\Phi_i}$ with $L_{\max}$ denoting the maximum packet size (in bits). If the vector $\{\bar{D}_1, \bar{D}_2, \ldots, \bar{D}_N\}$ is schedulable (in virtual time) under a non-preemptive policy, then necessarily for any time interval $[v_0, v(t)]$ where v_0 and $v(t)$ are virtual times measured in bits,

$$L_{\max} \le \bar{D}_1 \tag{1}$$

$$\begin{aligned}\sum_{i=1}^{N} \{ & (L_{\max} + r_i \min(v(t) - v_0 - L_{\max} + \varrho, (v(t) - v_0 - \bar{D}_i))) \\ & \cdot I(\min(v(t) - v_0 - L_{\max} + \varrho, (v(t) - v_0 - \bar{D}_i)))\} + L_{\max} \\ & \le v(t) - v_0, \quad L_{\max} \le v(t) - v_0 < \bar{D}_N \end{aligned} \tag{2}$$

$$\begin{aligned}\sum_{i=1}^{N} \{ & L_{\max} + r_i \min(v(t) - v_0 - L_{\max} \\ & + \varrho, v(t) - v_0 - \bar{D}_i)\} \le v(t) - v_0, \quad v(t) - v_0 \le \bar{D}_N \end{aligned} \tag{3}$$

where $I(\cdot)$ is the indicator function, with $I(t) = 1$ if $t \ge 0$ and 0 otherwise.

Proof: Note that the arguments in the proof of Lemma 1 (pp. 1526) of [14] apply here in virtual time domain. We have also incorporated the effect of the lookahead window ϱ. □

Lemma 4.2. Let $\bar{D}_1 \le \bar{D}_2 \le \ldots \le \bar{D}_N$ where $\bar{D}_i = \frac{L_{\max}}{\Phi_i}$ with $L_{\max}$ denoting the maximum packet size (in bits). Any vector $\{\bar{D}_1, \bar{D}_2, \ldots, \bar{D}_N\}$ that satisfies the constraints of Lemma 4.1 is schedulable under our algorithm in virtual time domain.

Proof: The proof translates the arguments for EDF in the real time domain (see Lemma 2 (pp. 1527) of [14]) into the virtual time domain. □

Based on the above lemmas, we conclude the following Theorem 4.1 on the schedulability condition of our algorithm.

Theorem 4.1. The schedulable region of the algorithm consists of the set of vectors which satisfy the constraints

$$[k+1]L_{\max} \leq \bar{D}_k - \sum_{i=1}^{k-1}[r_i \min(\bar{D}_k - L_{\max} + \varrho, \bar{D}_k - \bar{D}_i)], \quad 1 \leq k \leq N-1 \quad (4)$$

$$NL_{\max} \leq \bar{D}_N - \sum_{i=1}^{N-1}[r_i \min(\bar{D}_N - L_{\max} + \varrho, \bar{D}_N - \bar{D}_i)] \quad (5)$$

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

Corollary 4.1. Since $L_{\max} \leq \bar{D}_i, \forall i$, and $\varrho \geq 0$, it follows that the schedulability condition can be simplified to

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

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

This is exactly the schedulability condition for EDF policy.

Interestingly, the value of ϱ plays no role in the schedulability of flows, though it does have an impact on the order of the schedule.

4.1.2. Throughput Guarantees

We first establish a lower bound on the aggregate service (in bits) received by any flow during a virtual time interval $[v_1, v_2]$. For flow i, $W_i(v_1, v_2)$ denotes its aggregate service in bits during interval $[v_1, v_2]$.

Lemma 4.3. If flow i is backlogged during the interval $[v_1, v_2]$ where v_1, v_2 are measured in bits, then its throughput $W_i(v_1, v_2)$ satisfies:

1. if flow i did not receive service at v_1^- (where $v_1 - \epsilon < v_1^- < v_1$ for an arbitrary small $\epsilon > 0$), i.e., it becomes backlogged from idle at v_1, then
$$W_i(v_1, v_2) \geq r_i(v_2 - v_1) - L_{\max} - 2L_{\max}\frac{r_i}{\Phi_i} \quad (6)$$
2. if a packet from flow i has received its services v_1^-, then
$$W_i(v_1, v_2) \geq r_i(v_2 - v_1) - L_{\max} - L_{\max}\frac{r_i}{\Phi_i} \quad (7)$$

Remark 4.1. Intuitively, Part (2) of Lemma 4.3 states the following: the aggregate service that flow i receives during interval $[v_1, v_2]$, given by $r_i(v_2 - v_1)$, should complete at most by virtual time $v_2 + \frac{L_{\max}}{\Phi_i}$. This is true by considering the schedulability condition: if flow i is schedulable, all packets arrived by time v should finish service by time $v + \bar{D}_i$ where $\bar{D}_i$ is the maximum delay allowed for packets of flow i. As for the lower throughput bound, we need to subtract one additional packet which receives services at v_1.

Remark 4.2. Lemma 4.3 shows that we do achieve the decoupling of delay and throughput in the sense that the long-term throughput is mainly determined by the rate weight parameter r_i, not the delay weight parameter Φ_i.

Lemma 4.4. If all other flows have been continually backlogged over $[v_1^-, v_2]$, then the maximum throughput that flow i may receive during $[v_1, v_2]$ (where v_1 and v_2 are measured in bits) is given by

$$W_i(v_1, v_2) \leq r_i(v_2 - v_1) + L_{\max} \quad (8)$$

Based on the above two lemmas, we then establish the throughput bound in the real time domain.

Theorem 4.2. If a flow i is continually backlogged over a real time interval $[t_1, t_2]$, then its aggregate service (in bits) $W_i(t_1, t_2)$ is bounded by

$$W_i(t_1, t_2) \geq r_i C(t_2 - t_1) - r_i \sum_{i \in F} L_{\max} - L_{\max} - \alpha L_{\max}\frac{r_i}{\Phi_i} \quad (9)$$

where $\alpha = 1$ if flow i has been receiving services at t_1^-; and $\alpha = 2$ if no packet of flow i was served at t_1^-.

4.1.3. Delay Guarantees

Since the algorithm is an EDF in virtual time, it follows trivially that the maximum delay (measured in seconds) in virtual time domain is given by $\frac{L_{\max}}{C\Phi_i}$ for flow i. In the following, we will establish delay bound in the real-time domain.

Theorem 4.3. (New Queue Delay)

The new queue delay $d_i^{new\_queue}$ for flow i, defined as the difference between the departure time and arrival time of its head of line (HOL) packet, is given by [4]

$$d_i^{new\_queue} \le \frac{L_{\max}}{C}\left(\sum_{j\in F}^{j\neq i} r_j/\Phi_i\right) + \left(\sum_{j\in F} L_{\max}/C\right) \le \frac{L_{\max}}{C\Phi_i} + \left(\sum_{j\in F} L_{\max}/C\right) \quad (10)$$

where F denotes the set of all flows, and C is the server rate.

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

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

Theorem 4.4. (Maximum Packet Departure Time)
The departure time of packet p_i^j of flow i, denoted by PDT_i^j, is given by [5]

$$PDT_i^j \le EAT(p_i^j) + \frac{L_{\max}}{C}\left(\sum_{j\in F}^{j\neq i} r_j/\Phi_i\right) + \left(\sum_{j\in F} L_{\max}/C\right) \le EAT(p_i^j) + \frac{L_{\max}}{C\Phi_i} + \left(\sum_{j\in F} L_{\max}/C\right) \quad (12)$$

where F denotes the set of all flows.

Proof The proof is a combination of Lemma 4.3 and the arguments similar to [3]. □

Remark 4.3. As we can see from (10) and (12), the maximum packet delay bounds (10) and (12) are determined by the delay weight parameter Φ_i, not the rate parameter r_i. This also illustrates how the delay is decoupled from the throughput quantitatively. □

4.1.4. Fairness Guarantees
Based on Lemma 4.3 and Lemma 4.4, we can easily arrive at the following results on fairness index.

Theorem 4.5. (Long-term Fairness Index)
For a continually backlogged flow i, it achieves the following long-term throughput fairness index:

$$\lim_{v\to\infty} \frac{W_i(0,v)}{v} = r_i \quad (13)$$

Proof: From Lemma 4.3 and Lemma 4.4, we have the following inequality:

$$r_iv - L_{\max} - 2L_{\max}\frac{r_i}{\Phi_i} \le W_i(0,v) \le r_iv + L_{\max} \quad (14)$$

Divide all sides by v, we have

$$r_i - \frac{L_{\max}}{v} - \frac{2L_{\max}}{v}\frac{r_i}{\Phi_i} \le \frac{W_i(0,v)}{v} \le r_i + \frac{L_{\max}}{v}$$

The result follows readily by letting $v \to \infty$. □

Remark 4.4. It is easy to see that the result (13) still holds for a flow i that has been idle (over some finite time intervals) for a finite number of times.

Based on Lemma 4.3 and Lemma 4.4, we can easily conclude the following results on the fairness index adopted in the literature [3]:

Corollary 4.2. For any interval $[v_1, v_2]$ where v_1 and v_2 are virtual times measured in bits, in which flows i and j are continually backlogged during $[v_1, v_2]$, the difference in the service received by two flows is given by

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

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

Remark 4.5. As we can see from both Theorem 4.5 and Corollary 4.2, in addition to having larger schedulable region, our algorithm also provides fairness guarantees. This is a key difference between our algorithm and the real-time delay EDF policy. □

4.2. Performance of the Compensation Model in the Presence of Channel Errors

We describe the properties of our algorithm with the compensation model described in Section 3.4, in the presence of channel errors. We assume a fixed packet size L_p for all flows for simplicity of notation; Besides,

[4] For time-invariant weight parameters r_i and Φ_i, we can have better delay bound given by

$$d_i^{new\_queue} \le \frac{L_{\max}}{C}\left(\sum_{j\in F}^{j\neq i} r_j/\Phi_i\right) + \frac{L_{\max}}{C} \le \frac{L_{\max}}{C\Phi_i} + \frac{L_{\max}}{C}$$

This can be proved by using the packet-invariant-ordering property [14]. However, if r_i and Φ_i are time-varying, this does not hold in general.

[5] For time-invariant weight parameters r_i and Φ_i, we can have better delay bound given by

$$PDT_i^j \le EAT(p_i^j) + \frac{L_{\max}}{C}\left(\sum_{j\in F}^{j\neq i} r_j/\Phi_i\right) + \frac{L_{\max}}{C} \le EAT(p_i^j) + \frac{L_{\max}}{C\Phi_i} + \frac{L_{\max}}{C} \quad (11)$$

This can be again proved by using the packet-invariant-ordering property [14]. However, if r_i and Φ_i are time-varying, this does not hold in general.

as in Section 3.3, we define a flow i as *leading* over a time interval $[t_1, t_2]$ if its credit $C_i(t) := E(i) > 0$ (in bits) for all $t \in [t_1, t_2]$; as *lagging* over $[t_1, t_2]$ if its credit $C_i(t) := -G(i) < 0$ (in bits) for all $t \in [t_1, t_2]$; otherwise, we say flow i is *in-sync* if $C_i(t) = 0$ throughout $[t_1, t_2]$.

In the following, we provide throughput and delay guarantees for all three types of flows.

4.2.1. In-sync flows

For a flow i, if the channel is clean over $[t_1, t_2]$, the following result holds trivially from the operation of our algorithm:

Theorem 4.6. (Throughput and delay over clean channel)

For a flow which is in-sync at t_1 and has clean channels throughout the time interval $[t_1, t_2]$, its minimum throughput and maximum packet delay over $[t_1, t_2]$ are the same as those in the error-free algorithm.

If the channel is error prone over time $[t_1, t_2]$, the following result on HOL packet delay holds:

Theorem 4.7. (Delay on error-prone channels)

For a flow i, assume that it perceives no more than e_i error slots over any S_i slots. Then, for an in-sync flow i, the following results on maximum delay for HOL packets d_i^{HOL} hold:

1. If the channel plays the adversary, and the following error condition holds:

$$\frac{e_i}{S_i} < \frac{C}{L_p} r_i \tag{15}$$

then the maximum delay for HOL packet d_i^m is given by

$$d_i^m = (\sum_{j \in F} L_p/C) + d_i \tag{16}$$

where d_i is calculated by

$$1 + \frac{L_p}{L_p + \sum_{j \in \mathcal{G}} G_{\max}(j)} \sum_{k \in \mathcal{L}} \frac{C_k r_k C \Delta t / L_p}{r_k C \Delta t + C_i^{\max}} + \left(d_i - \frac{L_p}{C \min(\Phi_i, r_i)}\right) \frac{C}{L_p} r_i = \lceil \frac{d_i}{S_i} \rceil e_i + 1 \tag{17}$$

with Δt being given by $\Delta t = d_i - \frac{L_p}{C \min(\Phi_i, r_i)}$. An upper bound on d_i^m is given by

$$d_i^m \le (\sum_{j \in F} L_p/C) + \frac{e_i + 1 + \max(1, \frac{r_i}{\Phi_i})}{\frac{C r_i}{L_p} - \frac{e_i}{S_i}} \tag{18}$$

2. We take the worst-case percentage of channel errors as $\frac{e_i}{S_i}$. then an upper bound on the worst-case expected delay for HOL packets of flow i, given the worst-case channel error percentage as $\frac{e_i}{S_i}$, is given by

$$d_i^e \le (\sum_{j \in F} L_p/C) + \frac{\frac{1}{1 - e_i/S_i} - 1 + \max(1, \frac{r_i}{\Phi_i})}{C r_i / L_p} \tag{19}$$

Remark 4.6. The error condition given in (15) states that even if the channel plays the adversary, as long as the error rate is less than a flow's long term rate, the HOL packet will be delivered eventually.

Theorem 4.8. (Throughput on error-prone channels over $[t_1, t_2]$)

For a flow j, assume that the channel has no more than e_i error slots over any S_j slots. Then, for an in-sync flow i, its throughput is given by

$$\begin{aligned} W_i(t_1, t_2) \ge\ & r_i C(t_2 - t_1) - r_i \sum_{i \in F} L_{\max} - L_{\max} \\ & - \alpha L_{\max} \frac{r_i}{\Phi_i} - \lceil \frac{C(t_2 - t_1)}{S_i L_p} \rceil e_i L_p \\ & + \frac{L_p}{\sum_{j \in \mathcal{G}} G_{\max} + L_p} \sum_{k \in \mathcal{L}(t_1)} \frac{C_k(t_1) r_k C(t_2 - t_1)}{r_k C(t_2 - t_1) + C_i^{\max}} \end{aligned}$$

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

Proof: In the presence of channel errors, over a time window $[t_1, t_2]$, if the channel plays the adversary, whenever flow i is selected for transmission, the channel will assign an error slot for it. The maximum number of error slots is given by $\lceil \frac{C(t_2 - t_1)}{S_i L_p} \rceil e_i$. Besides, we can obtain the number of compensation slots that flow i gets, similar to Theorem 4.7. Then the throughput bound follows. □

4.2.2. Leading flows

As described before, the rate weight $r_i(t)$ for a leading flow i over $[t_1, t_2]$ is adjusted as follows:

$$r_i(t) = r_i \left(1 - \frac{C_i(t)}{C_i^{\max}}\right), \quad t \in [t_1, t_2] \tag{20}$$

where $C_i^{\max}$ is the maximum credit that flow i can accumulate, and $C_i(t)$ is the current credit of flow i at t, r_i is a constant.

For a leading flow i throughout $[0, t]$, the following long-term throughput guarantee is straightforward:

Theorem 4.9. (Long-term throughput guarantees over clean channels)

Consider a leading flow i over a time interval $[0, t]$. Assume that it is continually backlogged. Then its aggregate service in bits $W_i(0, t)$ is bounded by

$$W_i(0, t) \ge r_i C t - r_i \sum_{i \in F} L_{\max} - L_{\max} - L_{\max} \frac{r_i}{\Phi_i} + C_i(t)$$

However, over a short time period, a leading flow may give up its lead so that lagging flows can catch up. The question is how much of its service a leading flow should give up while the lagging flow is catching up. As manifested by the following theorem, the rate adaptation scheme (20) will result in graceful degradation for leading backlogged flows in the sense that it is guaranteed to receive a minimum fraction of services over any short time interval, that is, it will never be 'starved' over any short time period as long as it perceives a clean channel and is backlogged. This way, an adaptive flow may continue to run by eventually decreasing its quality.

Theorem 4.10. (Graceful service degradation for a leading flow)

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

$$C_i(t) = C_0 e^{-\frac{r_i C}{C_i^{\max}}(t-t_1)} , \qquad (21)$$

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

$$r_i(t) = r_i \left(1 - \frac{C_0}{C_i^{\max}} e^{-\frac{r_i C}{C_i^{\max}}(t-t_1)}\right) , \qquad (22)$$

its maximum graceful degradation time t_g is given by

$$t_g = t_1 + \frac{C_i^{\max}}{r_i C} \ln C_0 \qquad (23)$$

Remark 4.7. As we can see from (22) and (21), both the credit and instantaneous rate of a leading flow decreases exponentially. This is different from the degradation schemes in WPS [4] and CIF-Q [5].

Theorem 4.11. (Short-term throughput guarantee for a leading flow over clean channels)

Consider a leading flow i over a time interval $[t_1, t_2]$ which is continually backlogged. Then its aggregate service in bits $W_i(t_1, t_2)$ is bounded by

$$W_i(t_1, t_2) \geq r_i C(t_2 - t_1) - r_i \sum_{i \in F} L_{\max} - L_{\max}$$
$$-L_{\max} \frac{r_i}{\Phi_i} - r_i C \int_{t_1}^{t_2} \frac{C_i(t)}{C_i^{\max}} dt$$

Proof: During an instantaneous time interval $[t, t + dt]$, it follows from (20) that the maximum amount of service that a leading flow gives up for lagging flow compensation is given by $\frac{C_i(t)}{C_i^{\max}} r_i C dt$, where C is the server rate. Therefore, the aggregate amount of service that flow i gives up during $[t_1, t_2]$ is as:

$$\int_{t_1}^{t_2} \frac{C_i(t)}{C_i^{\max}} r_i C dt \qquad (24)$$

Then, by referring to (9) and noting that leading flow i is not considered to be idle until it gives up its lead, we conclude the proof. □

Corollary 4.3. Consider the case $C_i(t_1) = C_0$. Then the throughput $W_i(t_1, t_2)$ for a continually backlogged leading flow i over $[t_1, t_2]$ is bounded by

$$W_i(t_1, t_2) \geq r_i C(t_2 - t_1) - r_i \sum_{i \in F} L_{\max} - L_{\max}$$
$$-L_{\max} \frac{r_i}{\Phi_i} - C_0(1 - e^{-\frac{r_i C}{C_i^{\max}}(t_2 - t_1)})$$

Theorem 4.12. (Short-term throughput guarantee for a leading flow over error-prone channels)

Consider a leading flow i over a time interval $[t_1, t_2]$ which is continually backlogged. Assume that the channel has less than e_i error slots over any S_i slots. Then its aggregate service in bits $W_i(t_1, t_2)$ is bounded by

$$W_i(t_1, t_2) \geq r_i C(t_2 - t_1) - r_i \sum_{i \in F} L_{\max} - L_{\max}$$
$$-L_{\max} \frac{r_i}{\Phi_i} - r_i C \int_{t_1}^{t_2} \frac{C_i(t)}{C_i^{\max}} dt - \lceil \frac{C(t_2 - t_1)}{S_i L_p} \rceil e_i L_p$$

Since a leading flow is always sending packets ahead of its expected arrival time, the following result on delay bound holds trivially.

Theorem 4.13. (Delay bound for a leading flow)

As long as a flow i is leading over $[t_1, t_2]$, no matter whether the channel is clean or not, its maximum delay with respect to its expected arrival time is zero.

4.2.3. Lagging flows

Theorem 4.14. (Delay over clean channels) Consider a backlogged lagging flow i at t, which has a credit $C_i(t)$ (in bits). Assume that all flows have a uniform packet size L_p, and all have clean channels and are backlogged from time t, and $\sum_{j \in F} r_j = 1$ without any loss of generality, where F denotes the total set of flows. Then the HOL packet delay of flow i (defined as the difference between the departure time and its expected arrival time of the HOL packet) is upper bounded by

$$d_i^{HOL} := PDT_i^{HOL} - EAT(p_i^{HOL}) \leq$$
$$\left(\sum_{j \in F} L_p / C\right) + \frac{L_p}{C \Phi_i} + \frac{|C_i(t)|}{C r_i} + \min\left\{\frac{L_p}{r_i C}, T_m\right\} \quad (25)$$

where T_m is calculated by

$$C T_m \sum_{k \in \mathcal{L}(t)} \frac{C_k(t) r_k / L_p}{r_k C T_m + C_i^{\max}} = \frac{\sum_{j \in \mathcal{G}(t)} |C_j(t)|}{|C_i(t)|} \quad (26)$$

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

Corollary 4.4. If we take $C_i^{\max} = \mu r_i$ for some constant $\mu > 0$ for every flow $i \in F$, the time T_m of the above Theorem is given by

$$T_m = \frac{\mu}{C} \frac{\sum_{j\in\mathcal{G}(t)} |C_j(t)|}{(|C_i(t)|/L_p) \sum_{k\in\mathcal{L}(t)} C_k(t) - \sum_{j\in\mathcal{G}(t)} |C_j(t)|}$$

Using the same arguments, during a time interval $[t_1, t_2]$, if all flows have clean channels, a lagging flow i receives at least the following number of compensation slots:

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

Then the throughput bound for a lagging flow follows readily:

Theorem 4.15. (Throughput guarantees for lagging flows over clean channels)

Consider a backlogged lagging flow i at t_1, which has a credit $C_i(t_1)$ (in bits). Assume that all flows have clean channels over time interval $[t_1, t_2]$, and $\sum_{j\in F} r_j = 1$ without any loss of generality, where F denotes the total set of flows. Then, the minimum aggregate service (in bits) that flow i receives during time interval $[t_1, t_2]$, is given by

$$W_i(t_1, t_2) \geq r_i C(t_2 - t_1) - r_i \sum_{i\in F} L_{\max} - L_{\max} - L_{\max}\frac{r_i}{\Phi_i} + \min\Big\{|C_i(t_1)|, \sum_{k\in\mathcal{L}(t_1)} \frac{C_k(t_1) r_k C(t_2 - t_1)}{r_k C(t_2 - t_1) + C_i^{\max}} \frac{|C_i(t_1)|}{\sum_{j\in\mathcal{G}(t_1)} |C_j(t_1)|}\Big\}$$

Theorem 4.16. (Delay on error-prone channels)

For a lagging flow i, assume that it perceives less than e_i error slots over any S_i slots, and its minimum credit over $[t_1, t_2]$ is C_i^m. Then, the following results on maximum delay for HOL packets of flow i that are served during $[t_1, t_2]$ hold:

1. If the channel plays the adversary, and the following error condition holds:
$$\frac{e_i}{S_i} < \frac{C}{L_p} r_i \tag{27}$$
then the maximum delay for HOL packet d_i^m is given by
$$d_i^m = (\sum_{j\in F} L_p/C) + d_i \tag{28}$$
where d_i is calculated by
$$\frac{C}{L_p} r_i d_i + \frac{C_i^m}{C_i^m + \sum_{j\in\mathcal{G}} G_{\max}(j)} \sum_{k\in\mathcal{L}} \frac{C_k r_k C \Delta t/L_p}{r_k C\Delta t + C_i^{\max}} = \lceil \frac{d_i}{S_i} \rceil e_i + 1$$
with Δt being given by $\Delta t = d_i - \frac{L_p}{C r_i}$.
An upper bound on d_i^m is given by
$$d_i^m \leq (\sum_{j\in F} L_p/C) + \frac{e_i + 1}{\frac{C r_i}{L_p} - \frac{e_i}{S_i}}$$
2. We take the worst-case percentage of channel errors as $\frac{e_i}{S_i}$. then an upper bound on the worst-case expected delay for HOL packets of flow i, given the worst-case channel error percentage as $\frac{e_i}{S_i}$, is given by
$$d_i^e \leq (\sum_{j\in F} L_p/C) + \frac{L_p}{C r_i} \frac{1}{1 - e_i/S_i} \tag{29}$$

Proof: The proof follows the arguments similar to Theorem 4.7. □

Theorem 4.17. (Throughput guarantees for lagging flows over error-prone channels)

Consider a backlogged lagging flow i at t_1, which has a credit $C_i(t_1)$ (in bits). Assume that the channel has less than e_i error slots over S_j slots. Then, the minimum aggregate service (in bits) that flow i receives during time interval $[t_1, t_2]$, is given by

$$W_i(t_1, t_2) \geq r_i C(t_2 - t_1) - r_i \sum_{i\in F} L_{\max} - L_{\max} - L_{\max}\frac{r_i}{\Phi_i} - \lceil \frac{C(t_2 - t_1)}{S_i L_p} \rceil e_i L_p + \min\Big\{|C_i(t_1)|, \sum_{k\in\mathcal{L}(t_1)} \frac{C_k(t_1) r_k C(t_2 - t_1)}{r_k C(t_2 - t_1) + C_i^{\max}} \frac{|C_i(t_1)|}{\sum_{j\in\mathcal{G}(t_1)} |C_j(t_1)|}\Big\}$$

Proof: The arguments follow similar to that of Theorem 4.8.

5. Simulation Results

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

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

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

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

The wireless channel in our simulations evolves according to a two-state discrete Markov chain. 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

$$P_G = \frac{p_g}{p_g + p_e}\ , \qquad P_E = \frac{p_e}{p_g + p_e}. \tag{30}$$

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. Note that channel prediction is a pluggable module for the purpose of our simulations. Any other mechanism that predicts channel error efficiently can be easily incorporated in our simulations.

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

Example 1: Decoupling of rate and delay. We illustrate that our algorithm has a larger schedulable region than other proposed wireless fair queueing algorithms, due to the decoupling of delay and bandwidth. 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. We consider two scenarios :

Src	r_i	Φ_i	W	P_l	$D_{\max}$	D_{avg}	σ_D	d^{nq}
				Case I: Entire Run				
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
				Case II: Small time windows				
1	0.11	0.11	0.11	0	22.5	8.4	6.8	6.7
2	0.44	0.44	0.44	0	10.7	3.1	1.7	1.4
3	0.44	0.44	0.44	0	11.1	3.1	1.4	1.7

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

Src	r_i	Φ_i	W	P_l	$D_{\max}$	D_{avg}	σ_D	d^{nq}
				Case I: Entire Run				
1	0.11	0.9	0.11	0	37.5	1.0	2.7	16
2	0.44	0.09	0.44	0	40.8	2.9	4.4	23
3	0.44	0.009	0.44	0	64.7	6.8	7.4	30
				Case II: Small time windows				
1	0.11	0.9	0.11	0	5.4	0.8	1.4	4
2	0.44	0.09	0.44	0	11.8	2.6	2.9	5
3	0.44	0.009	0.44	0	20.1	6.6	5.1	7

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

a. In this scenario, we set the delay weights Φ_i to be equal to the rate weights r_i of the sources, and the lookahead $\varrho = \infty$. Our algorithm is now the same as the Wireless Fair Queueing algorithm [4]. The parameters and the simulation results over the entire run (I) and over small time windows (II) are given in Table 1. As expected, the rates obtained by the sources are proportional to their weight, and the configuration is schedulable.[6]

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

In comparison, IWFQ [4], CIF-Q [5] and CBQ-CSDPS [9] do not provide delay-bandwidth decoupling. Packet delays under these algorithms are tightly cou-

[6] 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. This is consistent with the analytical results in the previous section.

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

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

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

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

pled to the rate weights of the flows. SBFA [6] can achieve delay-bandwidth decoupling if its error-free service model can.

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

We consider three sources, where Sources 1 and 2 are Markov-modulated Poisson processes (MMPPs), 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 channel 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 $r_i = 0.333$. 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 retransmits to 8, i.e. a packet is dropped if it is not successfully transmitted after nine attempts. The simulation results are presented in Table 3. The simulation is done over an entire run (I), as well as over a set of small time windows (II). For purposes of comparison, we also present the simulation results for same set of flows with error-free channels (III). 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 large delays. All flows get the rates in proportion to their rate weights. From the results with the small time windows, we see that our compensation model provides for short term fairness and throughput guarantees even when the channels are error-prone with bounded error.

(b) Delay-sensitive flows: Instead of setting an upper limit on the number of retransmission attempts per packet, we set an upper limit on the maximum delay of a packet to be 150. If a packet is in the system for more than 100 time slots,then it is dropped; this could possibly happen even before it reaches the head-of-the-queue. Thus, our sources are now delay-sensitive, rather than error-sensitive.

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

In existing literature, IWFQ [4], CBQ-CSDPS [9] and SBFA [6] do not provide any short-term bounds on fairness or throughput. CIF-Q [5] provides short-term fairness and throughput bounds in the average case, but it degenerates to IWFQ in the worst case.

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 50. Figure 3 presents the plot of the number of packets served over time.

We are able to see that up to time t=100, Flow 1 is starved as it encounters channel error. Flow 2 receives the excess service. Due to the inherent nature of our algorithm that avoids disturbing in-sync flows as much as possible, we see that Flow 3 does not receive any of the excess service. At time $t = 100$, Flow 1 has accumulated a lag of 50, Flow 2 has a lead of 50, while Flow 3 does not have any lead or lag. After t=100, Flow 1 experiences a clean channel and it starts giving up its lag. 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 credit, it does not get starved, and observes a graceful degradation of service, while Flow 3 is not disturbed at all. Flow 1 also sees a graceful decrease of rate to 0.33

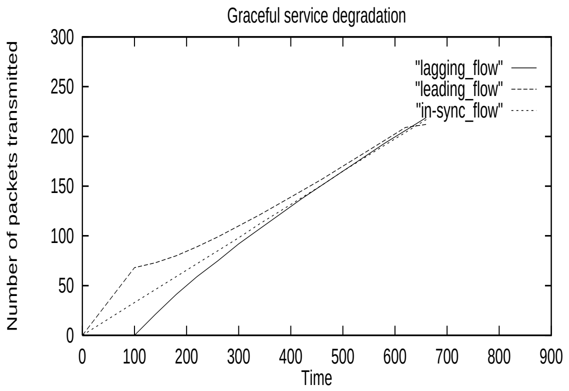

Figure 3. Graceful Service Degradation

	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

as it reduces its lag to 0.

In Idealized Wireless Fair Queueing [4], from time t = 100, the curve would have levelled off for the leading flow and the in-sync flow, since the lagging flow would have had a higher precedence with its lower tags. This would happen till the lagging flow caught up with the leading and in-sync flows. CBQ-CSDPS [9] will behave in a similar way.

SBFA [6] compensates for the lagging flow from t=100 using the slots in the Long-Term Fairness Server (LTFS). There will be no in-sync flow in this case, since the other two flows receive excess service and become leading. From t=100, the lagging flow will get twice the bandwidth than the other flows, resulting in a linear service degradation for the leading flows. CIF-Q [5] uses the parameter α to control how soon a leading flow gives up its lead, and the service degradation achieved is linear. However, in the worst case, the service degradation for both SBFA and CIF-Q will degenerate to IWFQ. In our algorithm, we use the parameter $E_{\max}(i)$ to determine how a leading flow gives up its lead, and we achieve exponential degradation of service even in the worst case, which is more graceful.

Example 4: Adaptive Sources. A delay-sensitive flow that drops its packets when they exceed their delay bounds (due to channel error) will cease to be backlogged and thus lose its compensation. A flow can react to this packet dropping by generating additional packets equal to the number of packets lost at a higher rate.

In this example, we look at 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 20 implies that when a source generates excess packets in reaction to a packet dropping, it will be 20 time

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

Table 6
Effect of adaptive nature of source on throughput

units after the dropping 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 decrease in throughput is observed. Let us consider three flows : Flow 1 has an error-free channel at all times, Flow 2 and Flow 3 have channel error according to Table 5. All flows are MMPP sources with $\lambda_i = 1.2$. All the flows are delay-sensitive with the delay bound = 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 with smaller time-windows, i.e. when Flow 3 becomes more adaptive with respect to the rate. We see a 2 % 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 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 for supporting bounded delay access (and hence guaranteed QoS) to a shared wireline link, wireline fair queueing algorithms cannot be applied directly to the wireless domain because they do not address issues of location-dependent and bursty channel error, channel contention, dynamic channel capacity, shared broadcast channels for uplink and downlink flows, etc.

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

References

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

[2] A. Demers, S. Keshav and S. Shenker, "Analysis and simulation of a fair queueing algorithm," *Proc. of ACM SIGCOMM'89*, Austin, TX, September 1989.

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

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

[5] T.S. Eugene Ng., I. Stoica and H. Zhang, "Packet fair queueuing algorithms for wireless networks with location-dependent errors," *Proc. of IEEE INFOCOM'98*, San Francisco, CA, March 1998.

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

[7] P. Bhagwat, P. Bhattacharya, A. Krishma and S. Tripathi, "Enhancing throughput over wireless LANs using channel state dependent packet scheduling," *Proc. of IEEE INFOCOM'96*, San Francisco, CA, March 1996.

[8] C. Chang, J. Chang, K. Chen and M. You, "Guaranteed quality-of-service wireless access to ATM," *Proc. of IEEE GLOBECOM'96*, London, UK, November 1996.

[9] M. Srivastava, C. Fragouli, and V. Sivaraman, "Controlled Multimedia Wireless Link Sharing via Enhanced Class-Based Queueing with Channel-State-Dependent Packet Scheduling," *Proc. of IEEE INFOCOM'98*, San Francisco, CA, March 1998.

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

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

[12] S. Ha, K.W.Lee and V. Bharghavan, "Performance evaluation of scheduling algorithms in integrated services packet networks," *Proc. of ISCC'98*, Athens, Greece, July 1998.

[13] M.J. Karol, Z. Liu, and K.Y. Eng. "An efficient demand-assignment multiple access protocol for wireless packet (ATM) networks," *ACM Journal on Wireless Networking*, Vol. 1, No. 3, pp. 269-279, December 1995.

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

7. Appendix

Proof of Lemma 4.3: We prove Part (2) first. Consider the nontrivial case when $v_2 \geq v_1 + \frac{L_{\max}}{r_i} + \frac{L_{\max}}{\Phi_i}$. We first prove that there is a packet from flow i that is served in $[v_1, v_2]$. For this case, since a packet from flow i has received its services at v_1^-, denoted it as p_i^{k-1}, therefore, $S(p_i^{k-1}) \leq v_1$ and $F(p_i^{k-1}) \geq v_1$. Now consider packet p_i^k. From the tagging scheme, the tags of packet p_i^k are computed as

$$S(p_i^k) = S(p_i^{k-1}) + \frac{L_i^{k-1}}{r_i}, \quad F(p_i^k) = S(p_i^k) + \frac{L_i^k}{\Phi_i}$$

It is easy to see that $S(p_i^k) \leq v_1 + \frac{L_{\max}}{r_i} < v_2$ and $v_1 < F(p_i^k) \leq v_1 + \frac{L_{\max}}{r_i} + \frac{L_{\max}}{\Phi_i} \leq v_2$. Hence, there is a packet p_i^k that is served in $[v_1, v_2]$.

Now let us denote the last packet from flow i to receive services in $[v_1, v_2]$ as p_i^{k+n}, then it is easy to see that $F(p_i^{k+n+1}) \geq v_2$. Using the tagging scheme of Section 3.1, the following holds

$$\begin{aligned} S(p_i^{k+n}) &= \left(F(p_i^{k+n+1}) - \frac{L_i^{k+n+1}}{\Phi_i}\right) - \frac{L_i^{k+n}}{r_i} \\ &\geq v_2 - \frac{L_{\max}}{\Phi_i} - \frac{L_i^{k+n}}{r_i} \end{aligned}$$

Using the tagging scheme of Section 3.1, we can derive

$$S(p_i^{k+n}) - S(p_i^k) = \sum_{j=0}^{n-1} \left(\frac{L_i^{k+j}}{r_i}\right) \tag{31}$$

Then, it follows that

$$\sum_{j=0}^{n-1} \left(\frac{L_i^{k+j}}{r_i}\right) \geq v_2 - \frac{L_{\max}}{\Phi_i} - \frac{L_i^{k+n}}{r_i} - \left(v_1 + \frac{L_{\max}}{r_i}\right)$$

Hence, we obtain the following inequality:

$$\sum_{j=0}^{n} \left(\frac{L_i^{k+j}}{r_i}\right) \geq v_2 - v_1 - \frac{L_{\max}}{\Phi_i} - \frac{L_{\max}}{r_i} \tag{32}$$

Since $W_i(v_1, v_2) \geq \sum_{j=0}^{n} L_i^{k+j}$, then it follows that

$$W_i(v_1, v_2) \geq r_i(v_2 - v_1) - L_{\max} - L_{\max}\frac{r_i}{\Phi_i}$$

This completes the proof of Part (2).

For Part (1), flow i started from idle and becomes backlogged at time v_1. Then its head of line packet is tagged as $F(p_i^{k-1}) = V(A(p_i^{k-1})) + \frac{L_i^{k-1}}{\Phi_i} \leq v_1 + \frac{L_i^{k-1}}{\Phi_i}$. In the worst-case scenario, all other flows have been continually backlogged before v_1, therefore, flow i has to wait for a maximum time $\frac{L_i^{k-1}}{\Phi_i}$ to receive its services. Therefore, the maximum services it lost during $[v_1, v_1 + \frac{L_i^{k-1}}{\Phi_i}]$ is bounded[7] by $r_i L_{\max}/\Phi_i$.

[7] In the normal case, an extra packet should be considered to obtain an upper bound. However, note that Part (2) has already considered it. Therefore, we do not need to add one extra packet here.

After the first packet has received service, the arguments for Part (2) still hold. Hence, its minimum throughput is given by

$$W_i(v_1, v_2) \geq r_i(v_2 - v_1) - L_{\max} - L_{\max}\frac{r_i}{\Phi_i} - r_i\frac{L_{\max}}{\Phi_i}$$

This completes the proof of Part (1). □

Proof of Lemma 4.4: The set of flow i packets during time interval $[v_1, v_2]$ have start tags at least v_1 and at most v_2. This set can be partitioned into two subsets:

- Set A which consists of packets that have start tag at least v_1 and strictly less than v_2, that is,

$$A = \{k \mid v_1 \leq S(p_i^k) \quad \text{and} \quad S(p_i^k) < v_2\} \quad (33)$$

- Set B which consists of packets whose start tags may be at most v_2, and whose finish tag is strictly larger than v_2. It is obvious that at most one packet belongs to this set.

For packets in Set A, using the equation (31), we can easily obtain

$$W_i(v_1, v_2) \leq r_i(v_2 - v_1) \quad (34)$$

Since at most one packet may belong to set B, it follows that

$$W_i(v_1, v_2) \leq r_i(v_2 - v_1) + L_{\max} \quad (35)$$

This concludes the proof of Lemma 4.4. □

Proof of Theorem 4.2: We will derive a minimum throughput for flow i during real time interval $[t_1, t_2]$, using arguments similar to those of [3]. The minimum throughput for a backlogged flow is achieved when all other flows are continually backlogged as we considered in Lemma 4.3 and Lemma 4.4. We choose $v(t_1) = v_1$ and denote the aggregate length of packets from all flows to receive service in some virtual time interval $[v_1, v_2]$ (measured in seconds) as $\bar{W}_i(v_1, v_2)$. From Lemma 4.4, we have

$$\bar{W}(v_1, v_2) \leq \sum_{i \in F} r_i C(v_2 - v_1) + \sum_{i \in F} L_{\max} \quad (36)$$

by noting that the virtual time in (8) is measured in bits, which can be easily translated into seconds assuming a constant server rate C.

Since $\sum_{i \in F} r_i = 1$, we have

$$\bar{W}(v_1, v_2) \leq C(v_2 - v_1) + \sum_{i \in F} L_{\max} \quad (37)$$

We define the virtual time v_2 as

$$v_2 = (t_2 - t_1) + v_1 - \frac{\sum_{i \in F} L_{\max}}{C}$$

Substitute v_2 into $\bar{W}(v_1, v_2)$, we have

$$\bar{W}(v_1, v_2) \leq C(t_2 - t_1) \quad (38)$$

Let $\bar{t}_2$ be such that $v(\bar{t}_2) = v_2$. Therefore, The real time it takes the server to finish serving $\bar{W}(v_1, v_2)$, given by $\bar{t}_2 - t_1$, satisfies:

$$\bar{t}_2 - t_1 = \frac{\bar{W}(v_1, v_2)}{C} \leq \frac{C(t_2 - t_1)}{C} \quad (39)$$

Hence, it follows that $\bar{t}_2 \leq t_2$.

Therefore, it follows that $W_i(t_1, t_2) \geq W_i(t_1, \bar{t}_2) = W_i(v_1, v_2)$. Then, the results can be readily obtained from Lemma 4.3. □

Proof of Theorem 4.3: We present the outline of the proof due to lack of space. For new queue delay, the worst-case scenario happens in the following context: flow i starts to become backlogged and all other flows have been served and remain backlogged. Based on our tagging scheme, the maximum time (in bits) flow i has to wait is given by $\frac{L_{\max}}{\Phi_i}$. Therefore, flow $j \neq i$ can receive services (before the head-of-line packet of flow i) bounded by at most $r_j \frac{L_{\max}}{\Phi_i} + L_{\max}$ where we take account at most one additional packet to be served. Therefore, the maximum aggregate length of packets from other flows, which are served before the HOL packet of flow i, is given as

$$\sum_{j \in F}^{j \neq i} \left(r_j \frac{L_{\max}}{\Phi_i} + L_{\max} \right)$$

By noting that the server has a rate C, and the maximum length of the HOL packet of flow i is $L_{\max}$, then it follows that its departure time T_i^{HOL} is bounded by

$$T_i^{HOL} \leq A_i^{HOL} + \frac{1}{C} \left\{ \sum_{j \in F}^{j \neq i} \left(r_j \frac{L_{\max}}{\Phi_i} + L_{\max} \right) + L_{\max} \right\}$$

where A_i^{HOL} is the arrival time of its head of line packet. This concludes the proof. □

Proof of Theorem 4.7: For a HOL packet of flow i, it might be selected for transmission due to one of the two reasons: it gets a normal slot due to its normal rate share r_i, or it gets a compensation slot since it is lagging. On a clean channel over a time window d_i, due to its rate share r_i and delay weight Φ_i, flow i gets the number of transmission slots s_i^t as

$$s_i^t := 1 + \left(d_i - \frac{L_p}{C \min(\Phi_i, r_i)} \right) \frac{C}{L_p} r_i \quad (40)$$

Now we consider the compensation slots that flow i can receive from leading flows. For a leading flow j to generate K_j compensation slots during a time window d_i, according to the rate adaptation scheme (20), the compensation rate is lower bounded by

$$\frac{C_j(t) - K_j L_p}{C_j^{\max}} r_j \quad (41)$$

Hence, the compensation slots that a leading flow j may generate during d_i satisfies: slots

$$K_j = \frac{C_j(t) - K_j L_p}{C_j^{\max}} r_j \frac{C\Delta t}{L_p} \tag{42}$$

Hence, the compensation slots K_j are given by

$$K_j = \frac{C_j(t) r_j C\Delta t / L_p}{C_j^{\max} + r_j C\Delta t} \tag{43}$$

A lagging flow i can receive at least the number of compensation slots:

$$\frac{L_p}{L_p + \sum_{j\in\mathcal{G}} G_{\max}(j)} \sum_{k\in\mathcal{L}} K_k \tag{44}$$

It follows that the minimum number of compensation slots s_i^c that flow i can receive is given by

$$s_i^c = \frac{L_p}{L_p + \sum_{j\in\mathcal{G}} G_{\max}(j)} \sum_{k\in\mathcal{L}} \frac{C_k r_k C\Delta t / L_p}{r_k C\Delta t + C_i^{\max}}, \quad \Delta t = d_i - \frac{L_p}{C\min(\Phi_i, r_i)}$$

Therefore, in case of clean channels, the number of clean slots that flow i can get during time d_i is given by $s_i^t + s_i^c$. In the presence of channel errors, to get at least one clean slots, then the following holds:

$$s_i^t + s_i^c = \lceil \frac{d_i}{S_i} \rceil e_i + 1 \tag{45}$$

Hence, the HOL packet delay d_i satisfies:

$$1 + \frac{L_p}{L_p + \sum_{j\in\mathcal{G}} G_{\max}(j)} \sum_{k\in\mathcal{L}} \frac{C_k r_k C\Delta t / L_p}{r_k C\Delta t + C_i^{\max}} + \left(d_i - \frac{L_p}{C\min(\Phi_i, r_i)}\right) \frac{C}{L_p} r_i = \lceil \frac{d_i}{S_i} \rceil e_i + 1 \tag{46}$$

where $\Delta t = d_i - \frac{L_p}{C\min(\Phi_i, r_i)}$.

To obtain an upper bound for the solution to (46), we ignore the compensation slots, i.e. $s_i^c = 0$, then an upper bound on the maximum delay d_i is given by

$$d_i \le \frac{e_i + 1 + \max(1, \frac{r_i}{\Phi_i})}{\frac{Cr_i}{L_p} - \frac{e_i}{S_i}} \tag{47}$$

where we need the following necessary condition to hold:

$$\frac{e_i}{S_i} < \frac{Cr_i}{L_p} \tag{48}$$

Note that in case when all HOL packets have identical tags, the maximum time for flow i to wait until its HOL packet has been selected is given by $\sum_{i\in F} L_p / C$, then the result (16) follows.

If we take the worst-case percentage of channel errors as $\frac{e_i}{S_i}$, then we should get at least one clean slot in the presence of errors so that the HOL packet can be served. That is,

$$(s_i^t + s_i^c)(1 - \frac{e_i}{S_i}) \ge 1 \tag{49}$$

Then, following similar procedures as in Part (1), by ignoring the compensation slots, an upper bound on the expected HOL packet delay, given the worst case percentage of channel errors as $\frac{e_i}{S_i}$, is given by

$$d_i \le \frac{\frac{1}{1 - e_i/S_i} - 1 + \max(1, \frac{r_i}{\Phi_i})}{Cr_i / L_p} \tag{50}$$

Then Part (2) follows readily. □

Proof of Theorem 4.10: From (20), it is easy to see that

$$-dC_i(t) = \frac{C_i(t)}{C_{\max}} r_i C dt\ , \qquad C_i(t_1) = C_0 \tag{51}$$

By integration, we arrive at

$$C_i(t) = C_0 e^{-\frac{r_i C}{C_i^{\max}}(t - t_1)}\ , \tag{52}$$

Substitute it into the equation (20), we obtain the instantaneous rate adaptation (22). To derive the maximum graceful degradation time t_2, we take the credit $C_i(t_2)$ as 1 bit in practice (a real packet should be at least in the order of byes), hence,

$$C_i(t_2) = C_0 e^{-\frac{r_i C}{C_i^{\max}}(t_2 - t_1)} = 1 \tag{53}$$

A simple calculation leads to inequality (23). This concludes the proof. □

Proof of Theorem 4.14: Let us first consider the maximum time that flow i needs to wait from time t to receive a clean slot according to our algorithm. Since flow i has a lagging credit $C_i(t)$ at time t, according to our algorithm, the first slot to be generated for flow i will be at the time

$$\min\left\{\frac{L_p}{r_i C}, T\left(\sum_{j\in\mathcal{G}(t)} |C_j(t)|/|C_i(t)|\right)\right\} \tag{54}$$

where $T(k)$ denotes the time it takes for leading flows to generate k slots, and $\mathcal{G}(t)$ denotes the set of lagging flows at t.

For a leading flow j to generate K_j compensation slots, according to the rate adaptation scheme (20), the compensation rate is lower bounded by

$$\frac{C_j(t) - K_j L_p}{C_j^{\max}} r_j \tag{55}$$

Hence, the time it takes a leading flow j to generate K_j compensation slots is given by

$$T(K_j) < T_m =: \frac{L_p}{C} \frac{C_j^{\max}}{(C_j(t) - K_j L_p) r_j} K_j \tag{56}$$

For lagging flow i to receive a compensation slot, K_j is subject to the constraint

$$\sum_{j \in \mathcal{L}(t)} K_j = \frac{\sum_{j \in \mathcal{G}(t)} |C_j(t)|}{|C_i(t)|} \tag{57}$$

where $\mathcal{L}(t)$ denotes the set of leading flows at t.

From equations (56) and (57), it is easy to derive that

$$CT_m \sum_{k \in \mathcal{L}(t)} \frac{C_k(t) r_k / L_p}{r_k CT_m + C_i^{\max}} = \frac{\sum_{j \in \mathcal{G}(t)} |C_j(t)|}{|C_i(t)|} \tag{58}$$

Hence, we arrive at an upper bound for the time that flow i needs to wait from time t to receive a clean slot.

As for the time that the HOL packet of flow i has waited until t since its expected arrival time, it is straightforward to apply the same arguments as those in the proof of Theorem 4.4. Thus we arrive at the result given by (25). □

Biographies

Songwu Lu 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 developing wireless scheduling and medium access protocols that provide quality of service.

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.

Vaduvur Bharghavan is an Assistant 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.