Fair Real-time Traffic Scheduling over A Wireless LAN *

Maria Adamou, Sanjeev Khanna, Insup Lee, Insik Shin, Shiyu Zhou
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104
{adamou, sanjeev, lee, ishin, shiyu}@saul.cis.upenn.edu

Abstract

Unpredictable wireless channel errors may cause applications with real-time traffic to receive degraded quality of services due to packet losses. In the presence of such errors, a challenging problem is how to schedule packets to achieve fairness among real-time flows and to maximize the overall system throughput simultaneously. We capture fairness by minimizing the maximum degradation in service over all flows. In this paper, we show that no online algorithm can guarantee a bounded performance ratio with respect to the optimal algorithm. We then compare four different online algorithms and evaluate them using simulations. The first two are EDF (Earliest Deadline First) and GDF (Greatest Degradation First) that consider only one aspect of our scheduling goal respectively. EDF is naturally suited for maximizing throughput while GDF seeks to minimize the maximum degradation. The next two are algorithms, called EOG (EDF or GDF) and LFF (Lagging Flows First), that consider the two aspects of our scheduling goal. EOG simply combines EDF and GDF, whereas LFF tries to favor lagging flows in a non-trivial manner. Our simulation results show that LFF is almost as good as EDF in maximizing the throughput and also is better than GDF in minimizing the maximum degradation. Finally, we also show that there is an optimal polynomial time algorithm for the offline version of the problem.

1 Introduction

Wireless communication technology has gained widespread acceptance in recent years. The IEEE 802.11 standard [12] has led wireless local area networks (LANs) into greater use. To provide bandwidth beyond the original 2 Mbps, the IEEE 802.11a [13] and 802.11b [14] supplements increase the bandwidth to 45 Mbps and 11 Mbps, respectively. With such high bandwidth, the demand for supporting multiple time-sensitive high-bandwidth traffic applications, such as video-on demand and interactive multimedia, in wireless LANs has been increasing. Thus, it is important to provide fair bandwidth allocation to multiple real-time traffics in a wireless LAN.

Many real-time packet scheduling and fair packet scheduling algorithms have been developed for wired networks. However, it is not clear how well these algorithms work for wireless networks, since wireless channels are subject to unpredictable *location-dependent* and *bursty* errors. In the presence of such unpredictable errors, real-time traffic applications can not fully utilize channel bandwidth assigned to them. A real-time traffic application may fail to send or receive some of its real-time packets in time due to channel errors. When experiencing packet losses above some specified thresholds, the application undergoes degraded quality of service (QoS). In this paper, we investigate the problem of fair scheduling of real-time packets with deadline constraints over wireless LAN, aiming at both achieving fair degradation and maximizing the system throughput.

There has been previous work on providing QoS guarantees over wireless links using call admission and scheduling [5, 6, 8]. While this previous work focused on providing service guarantees, it did not consider the issue of fairness or degradation. Several approaches have been introduced to deal with real-time task scheduling problem considering degraded QoS in an overload situation, where the system cannot meet the deadlines of all tasks. The notion of (m, k)-firm deadlines was introduced in [11] to represent less stringent guarantees for temporal constraints. In the (m, k)-firm deadlines model, it is adequate to meet the deadline constraints of m out of any k consecutive instances of a task in the overload condition. The *imprecise computation* [7, 18] and *IRIS (Increased Reward with Increased Service)* [9, 2] models

*This research was supported in part by NSF CCR-9988409, NSF CISE-9703220, ARO DAAG55-98-1-0393, ARO DAAG55-98-1-0466, and ONR N00014-97-1-0505 (MURI).

0-7695-1420-0/01 $17.00 © 2001 IEEE

were proposed to provide minimal quality of service in the overload condition. In these models, each task consists of a mandatory subtask and an optional subtask. The mandatory subtask should be completed before its deadline in order to provide minimal quality of service. The optional subtask can be executed before its deadline in order to enhance the quality of service after the completion of its corresponding mandatory subtask if there are enough resources in the system that are not committed to execute mandatory subtasks of other tasks. The execution of an optional subtask is associated with error in the imprecise computation model or reward in the IRIS model. The longer the optional subtask executes, the smaller the error, or the higher the reward. A typical scheduling objective is to ensure that the mandatory parts of all tasks are completed by their deadlines while the total errors/rewards in the system are minimized/maximized. These models addressed real-time scheduling problem with QoS degradation issues while optimizing a system-wide performance measure (error or reward). However, the issue of fairness was not addressed in these models.

Significant research efforts have been made to adapt packet fair scheduling algorithms to a wireless domain taking care of the wireless channel error characteristics [15, 16, 17]. These packets, however, are assumed to have no deadline constraints. In addition, there has been previous study on the issue of temporal fairness in periodic real-time scheduling. The notion of *pfairness* was introduced in [3] to minimize the length of time during which a task is denied service. This fairness notion enforces scheduling each periodic task proportional to its temporal property, which is the ratio of its execution time requirement to its period. A more recent study [20] addressed real-time scheduling problem taking fair degradation into account. This study used a finite range (window) to keep track of packet losses of multimedia streams and employed a scheduling policy that mainly worked on the greatest degradation (loss) first basis. While a fairness issue with deadline constraints was addressed in these two studies, an optimization issue of a system-wide performance measure such as the system throughput was not considered together.

In this paper, we address the scheduling problem of achieving fairness among real-time flows with deadline constraints as well as maximizing the throughput of all the real-time flows over a wireless LAN. To achieve fairness, we choose the scheduling objective of minimizing the maximum degree of the degraded QoS among all applications. Simultaneously, we try to maximize the overall system throughput. We show that for the problem of throughput maximization alone, there exists a simple online algorithm that achieves a performance ratio of two with respect to the optimal. However, for the problem of achieving fairness, we show that no online algorithm can guarantee a bounded performance ratio with respect to the optimal. Thus, no online algorithm can ensure a bounded performance ratio for the combined objectives. In contrast to the online scheduling problem with unpredictable channel errors, if all the errors are assumed to be known in advance, we show that there is a polynomial time offline scheduling algorithm that optimally achieves our scheduling objectives.

We then study four natural online algorithms and evaluate their performance using simulations. The first two are EDF (Earliest Deadline First) and GDF (Greatest Degradation First) that consider only one aspect of our scheduling goal respectively. EDF is naturally suited for maximizing throughput while GDF seeks to minimize the maximum degradation. The next two are algorithms, called EOG (EDF or GDF) and LFF (Lagging Flows First), that consider the two aspects of our scheduling goal. EOG simply combines EDF and GDF, whereas LFF tries to favor lagging flows in a more sophisticated manner. Our simulation results show that the LFF is almost as good as the EDF for maximizing the throughput and that it is the best for minimizing the maximum degradation.

This paper is organized as follows. Section 2 describes our scheduling model, including parameters and objectives. Section 3 provides theoretical results mentioned above. Section 4 presents the four online scheduling algorithms and the error handling mechanism that is used by our online algorithms. Section 5 presents the comparative evaluation of the four online algorithms based on simulation results. Section 6 summarizes the paper and identifies future work.

2 Scheduling Model

2.1 Real-Time Traffic Scheduling

We consider real-time traffic as *isochronous* (or synchronous) traffic that consists of message streams that are generated by their sources on a continuing basis and delivered to their respective destinations on a continuing basis. Such traffic includes periodic and sporadic messages that are characterized by stringent timing constraints. Periodic packets are generated at regular time intervals, and sporadic packets are generated at irregular intervals. The timing constraint of such a message is represented by a *deadline* that is the instant of time by which its delivery is required to be completed. The deadline is said to be *hard* if failure to meet it is considered as a fatal fault. The deadline is said to be *soft* if it is undesirable to miss it but a few misses of the deadlines

are tolerable. We refer to a message stream as a flow and a message instance as a packet.

In this paper, we will only consider the scheduling of periodic packets with soft deadlines. Examples of such packets include constant bit-rate (CBR) digitized voice and video data packets. We model a periodic soft real-time flow, f_i, as the tuple (v_i, D_i, e_i), $1 \leq i \leq N$. This means that each packet of flow f_i is generated with a period of v_i, each packet must be delivered to the destination within D_i units of time from its generation or arrival at the source; otherwise, the packet is lost, and a packet loss rate of up to e_i is acceptable. For simplicity, we assume that all the packets of f_i are of the same size L.

2.2 Cellular Wireless Network Model

We consider scheduling such real-time flows over a packet-switched wireless network that consists of multiple cells. Each cell is assumed to consist of a base station (BS) and multiple mobile hosts (MHs). Each flow is either an uplink (from a MH to the BS) or a downlink (BS to MH). The BS performs the scheduling of real-time packet deliveries using a polling scheme. The source of each flow f_i should notify the BS of a tuple (v_i, D_i, e_i) information prior to being scheduled. Each flow f_i is associated with a channel, ch_i, which is in one of two states, namely, *error state* or *error-free state*, at any time instant. At the moment when channel ch_i is in error state, flow f_i experiences a channel error. A packet delivery of flow f_i fails if the flow perceives the channel error at any time instant during the packet delivery.

2.3 Scheduling Performance Measure

We want to measure the performance of scheduling algorithms along two dimensions: from the points of view of the system and the user.

From the viewpoint of the system, it would be desirable to deliver as many packets as possible over all flows. We define the system throughput, T_{sys}, to be the fraction of delivered packets as follows:

$$T_{sys} = \frac{\sum_{i=1}^{N} M_i^a}{\sum_{i=1}^{N} M_i}, \tag{1}$$

where let M_i be the number of packets that flow f_i was supposed to deliver and let M_i^a be the number of packets that flow f_i actually successfully delivered.

From the viewpoint of the user, each user wants all periodic packets to be delivered on time. However, unpredictable wireless channel errors may cause the user with real-time traffic to receive degraded quality of services due to packet losses. We define a variable called, the degradation value, ϵ_i, to represent the degree of the degradation in quality of service. In other words, the degradation value is the distance between the desired quality of service and the quality of service being served. Each user naturally wants to minimize ϵ_i for its f_i. In order to be fair to all flows, we would like to minimize the maximum degradation value among all the flows. When ϵ_{max} is the maximum degradation value among all the flows, the smaller ϵ_{max} an scheduling algorithm generates, the fairer we consider the algorithm is. We consider a packet loss rate as a QoS parameter and define the degradation value ϵ_i for each flow f_i to be the distance between its acceptable packet loss rate e_i and the actual packet loss rate as follows:

$$\epsilon_i = 1 - \frac{M_i^a}{M_i} - e_i. \tag{2}$$

2.4 Scheduling Objective

In the presence of unpredictable errors, the problem is to determine which packets need to be scheduled, and in what order, so as to (a) minimize the maximum degree of degraded service among all flows, and (b) maximize the overall system throughput subject to (a). In other words, our scheduling objective is to determine the smallest ϵ_{max} for which we can ensure that $M_i^a \geq \lceil M_i(1 - e_i - \epsilon_{max}) \rceil$ and maximize $\sum_{i=1}^{N} M_i^a$.

3 Theoretical Results

We now establish some theoretical results that give an insight into the difficulty of finding good schedules in presence of errors. We start with some simple results concerning the worst-case behaviour of online algorithms for our problem. We measure the performance of any online algorithm in the competitive analysis framework of Sleator and Tarjan [19]. We say that an online algorithm is c-competitive if on any input sequence, it is guaranteed to produce a solution that is at least $1/c$ times as good as an optimal solution. Thus a 1-competitive algorithm gives essentially an optimal solution itself.

Proposition 1 *For any $\delta > 0$, there does not exist a $(2-\delta)$-competitive online algorithm for throughput maximization even when the system contains only two hosts.*

Proof. Consider a system with only two hosts, each one having one packet to transmit at some time t, with deadline $t+2$. If the online algorithm chooses the first time slot to schedule MH_1, an adversary generates an error in time slot 2 for MH_2, causing the online algorithm to loose its packet. On the other hand, if the online

algorithm first schedules the packet from MH_2, the adversary generates an error for MH_1 at time $t+1$. In either case, an optimal algorithm could have scheduled 2 packets. This process can be repeated indefinitely and thus the proposition follows.

In fact, the lower bound above is tight for throughput maximization.

Proposition 2 *There exists a 2-competitive online algorithm for throughput maximization.*

Proof. Consider the following online algorithm: at any time t, schedule any available packet from an error-free mobile host. Let P_{OPT} and P_{ON} denote the set of packets that are scheduled by the optimal offline algorithm and the online algorithm respectively. Moreover, let $P_{Both} = P_{OPT} \cap P_{ON}$ and $P_{Miss} = P_{OPT} \setminus P_{ON}$. Consider a packet $m \in P_{Miss}$ scheduled by the optimal at some time t. Since the online algorithm did not schedule this packet, it must be the case that it scheduled some other packet at time t. Therefore, $|P_{Miss}| \leq |P_{ON}|$. Putting together, $|P_{OPT}| = |P_{Both}| + |P_{Miss}| \leq 2 * |P_{ON}|$.

It fact, it is easy to see that any greedy online algorithm is 2-competitive for throughput maximization. However, once we take fairness into account, the situation becomes intractable.

Proposition 3 *For any $c \geq 1$, there does not exists a c-competitive algorithm for minimizing ϵ_{max} even when the system contains only two hosts.*

Proof. Consider the same scenario as described in the proof of Proposition 1. Assume that $\epsilon_1 = \epsilon_2 = 0$. Clearly, on the input sequence described there, an optimal algorithm achieves $\epsilon_{max} = 0$. On the other hand, since the online algorithm can be forced to miss half the packets, at least one of the flows has a degradation of $1/2$. Thus for any online algorithm, ϵ_{max} can be forced to be $1/2$. The ratio of these two quantities is unbounded and the proposition follows.

Given the difficulty of the online case even for the simpler goal of throughput maximization, a natural question to ask is if the problem remains intractable when the errors are all known in advance (i.e., an offline setting). We next show that this case is essentially equivalent to the maximum flow problem, well-known to be solvable efficiently in polynomial time (see [1], for instance).

Theorem 1 *There is a polynomial time offline algorithm that determines an optimal fair schedule with maximum throughput.*

Proof. We will reduce our problem to the maximum flow problem. For any $\epsilon > 0$, let $M_i^\epsilon = \lceil M_i(1 - e_i - \epsilon) \rceil$ where $1 \leq i \leq N$. Roughly speaking a schedule is said to be an (ϵ, α)-*schedule* if (i) it schedules at least M_i^ϵ packets for each flow i, and (ii) it sends at least $(\sum_{i=1}^N M_i^\epsilon) + \alpha$ packets overall. Our goal is to determine a pair (ϵ^*, α^*) such that (i) for any $\epsilon > \epsilon^*$ there does not exist an $(\epsilon, 0)$-schedule, and (ii) for any $\alpha > \alpha^*$, there does not exist an (ϵ^*, α)-schedule. We will construct an instance of the maximum flow problem for every candidate pair (ϵ, α) and output the solution corresponding to the best such pair found.

Let T denote $\sum_{i=1}^N M_i$, the total number of packets. It is easy to verify that there are only $O(\log T)$ candidate pairs need to be considered. We first use binary search over the set $\{1/T, 2/T, \ldots, 1\}$ to identify the smallest ϵ for which an $(\epsilon, 0)$-schedule exists. Then another binary search on the set $\{1, 2, ..., T\}$ determines the largest α for which this schedule stays feasible. Thus from here on, we assume without loss of generality that we know the optimal values of ϵ and α.

We construct a directed graph $G = (V, E)$ as follows. The vertex set of G contains two special vertices, namely a source s and a sink t, vertices $s_1, ..., s_N$ for each of the N flows, a vertex w, vertices $u_1, ..., u_p$ corresponding to the packets generated by the various flows, and vertices $v_1, ..., v_T$ corresponding to the various time slots that are available for scheduling packets. There is a directed edge from a vertex u_i to a vertex v_j if the packet corresponding to u_i could be scheduled at the jth time slot (i.e., the corresponding flow is in good state and the deadline of the packet has not yet expired). There is an edge from any vertex s_i to a vertex u_j if u_j corresponds to the packet generated by the ith flow. There are edges from each vertex v_j to the vertex t. All edges described thus far have a capacity of 1 on each edge. Finally, there are edges from the source vertex s to each of the s_i's as well as to the vertex w. The vertex w is connected to each u_i's by an edge of capacity 1. The directed edge (s, s_i) has capacity equal to M_i^ϵ and the edge (s, w) has capacity equal to α.

It is now an easy consequence of our construction that there exists an s-t flow of value $\sum_{i=1}^N M_i^\epsilon + \alpha$ if and only if there exists an (ϵ, α)-schedule.

4 Online Scheduling Algorithms

This section presents online scheduling algorithms and considers the issue of handling wireless channel errors in the context of real-time packet scheduling.

4.1 Online Scheduling Algorithms

We will denote by p_i^k the kth packet of a flow f_i. Each packet p_i^k is associated with an arrival (generation) time $A(p_i^k)$, at which it is ready to be transmitted. Since each flow f_i is periodic, $A(p_i^k) = A(p_i^{k-1}) + v_i$. Each packet p_i^k is also associated with a deadline $d(p_i^k)$, beyond which p_i^k cannot be scheduled for transmission. $d(p_i^k)$ is simply updated such that $d(p_i^k) = A(p_i^k) + D_i$. A packet is automatically dropped (lost) when its deadline expires (i.e., p_i^k is dropped at $d(p_i^k)$).

The scheduling goal of our scheduling model is to minimize ϵ_{max} and to maximize T_{sys} subject to the minimum ϵ_{max}. Under online scheduling, however, it is not feasible to maximize T_{sys} after finding the minimum ϵ_{max}. Hence, we slightly modify our scheduling goal for online scheduling algorithms such that our goal is (1) to minimize ϵ_{max} and (2) to maximize T_{sys} simultaneously. We consider the following online scheduling algorithms in order to achieve our scheduling goal.

4.1.1 Earliest Deadline First (EDF)

At any scheduling decision time, the packet with the earliest deadline is scheduled. In other words, at time t, a packet p_i^k is scheduled if $A(p_i^k) \leq t < d(p_i^k)$ and $d(p_i^k)$ is the minimum among all the packets. Ties are broken randomly. This algorithm is known to provide the maximum overall system throughput when there is no channel error. However, since this algorithm may starve some flows, it is not expected to perform well in minimizing ϵ_{max}.

4.1.2 Greatest Degradation First (GDF)

At any scheduling decision time, a packet belonging to the flow with the greatest degradation value is scheduled. At time t, a packet p_i^k is scheduled if $A(p_i^k) \leq t < d(p_i^k)$ and $\epsilon_i = \epsilon_{max}$. Ties are broken randomly. Intuitively, this algorithm is expected to result in minimizing ϵ_{max}. However, it is not suitable for maximizing the overall system throughput. For instance, we have two packets, p_i^k and $p_j^{k'}$, at time t such that $d(p_i^k) = t+1$ and $d(p_j^{k'}) = t+2$. We can schedule the two packets if p_i^k is scheduled at t and $p_j^{k'}$ is scheduled at $t+1$. If $\epsilon_j > \epsilon_i$, this algorithm schedules $p_j^{k'}$ at t. Then, it has no packet to schedule at $t+1$ since p_i^k is dropped at $t+1$.

4.1.3 EDF or GDF (EOG)

The EDF and the GDF are expected to perform well in one aspect of our scheduling goal respectively. However, they are not suitable to achieve simultaneously the two aspects of our scheduling goal. Thus, we consider a hybrid algorithm that attempts to combine positive aspects of the EDF and the GDF in expectation of performing well in the both aspects of our scheduling goal. At any scheduling decision time, if there is a packet whose deadline expires soon, packets are scheduled on the EDF basis. Otherwise, packets are scheduled on the GDF basis. At time t, if there is a packet p_i^k satisfying both $A(p_i^k) \leq t$ and $d(p_i^k) = t+1$, p_i^k is scheduled. Otherwise, it chooses a packet p_i^k satisfying $A(p_i^k) \leq t < d(p_i^k)$ and and $\epsilon_i = \epsilon_{max}$. Since this algorithm is a hybrid of the EDF and the GDF, it is expected to provide an average performance of the both.

4.1.4 Lagging Flows First (LFF)

In addition to the EOG algorithm, we consider another algorithm that simultaneously takes into account the two aspects of our scheduling goal. This algorithm uses a more refined approach than the EOG, and we show that it dominates EOG in both measures. The basic idea of the algorithm is to reserve time slots for packets to the latest possible time subject to meeting their deadlines in a decreasing order of flow degradation values. When a flow f_i is said to be *lagging* if its packet loss rate is above its acceptable loss rate e_i, this algorithms makes reservations from the most lagging flow to the least lagging flow. This idea of scheduling in the latest-ready-time-first order using in some priority order other than deadline is known to have a good performance under multiple scheduling constraints.

This scheduling algorithm consists of two parts: (1) when a new packet is generated or becomes available to be scheduled, this algorithm reserves a time slot for the packet if possible and maintains a reservation list, and (2) at any scheduling decision time, this algorithm schedules a packet according to the reservation list.

The priority in the reservation is a flow degradation value, favoring packets belonging to more lagging flows. A time slot is reserved for a new packet such that no time slot between the time slot and the deadline of the new packet is reserved for another packet that belongs to a less lagging flow than the new packet's flow. To find such a time slot, this algorithm performs a search process from the new packet's deadline to the current time looking for an empty time slot or an occupied time slot that is reserved for a packet belonging to a less lagging flow than the new packet's flow. When this search process reaches the current time without finding such a time slot, no time slot is reserved for the new packet and the new packet is not included in the reservation list. It if finds any empty slot, the slot is reserved for the new packet. If it finds a time slot associated with a less lagging flow during

```
Procedure ReserveTimeSlotForPacket(packet p) {
    /* This procedure reserves a time slot for packet p if possible */
    /* Let R be the array representing the reservation list */
    /* Let Q be the queue of packets for which time slots are not reserved */

    d = GetDeadline(p);
    while (d > 0) {
        if (R[d] = NULL) {      /* if time slot d is not reserved for any packet */
            R[d] = p; return;       /* time slot d is reserved for packet p */
        }
        else if (p.ε > R[d].ε) {      /* p.ε - degradation value of p*/
            Swap(p, R[d]);      /* p takes time slot d, and */
              /* the old packet R[d] needs to look for another available time slot */
        }
        d = d - 1;
    }
    InsertQueue(Q, p);      /* no time slot is reserved for p, and p is inserted into queue Q */
}

Function SchedulePacket() {
    /* This function returns a packet that is scheduled */
    int i = 1;
    while(R[i] = NULL) i++;
    p = R[i];      /* R[i] is chosen to be scheduled in the next time slot */
    R = AdjustArrayPointer (R);      /* R is adjusted such that R[i] = R[i + 1] */
    return p;
}
```

Figure 1: The LFF algorithm

the search, a reservation associated with the time slot is canceled and the time slot is newly reserved for the new packet. When an existing reservation canceled to yield a time slot for the new reservation, the search process continues until it reaches the current time looking for a new room for the packet associated with the canceled reservation. When the search process is done, the packet associated with the last canceled reservation is excluded from the reservation list. At any scheduling decision time, a packet that is given the earliest reservation time slot is scheduled.

Figure 1 presents the outline of the algorithm. `Procedure ReserveTimeSlotForPacket` is to reserve a time slot for a new packet, and `Function SchedulePacket` is to schedule a packet whose reservation time slot is the earliest in the reservation list. The scheduler maintains an one-dimensional array R representing the reservation list. $R[x]$ contains the packet p_i^k for which the xth time slot is reserved or *NULL* if the xth time slot is not reserved for any packet. We also consider a queue Q containing packets for which time slots are not reserved. When a packet p_i^k is generated, the algorithm decides whether or not the packet can be inserted into R. The scheduler looks for a time slot that is not reserved yet or is reserved to another packet $p_j^{k'}$ such that $\epsilon_j < \epsilon_i$ from its deadline to current time. If the scheduler finds an empty time slot u, the slot u is assigned to the packet p_i^k. If the scheduler finds a time slot v reserved for the packet $p_j^{k'}$ such that $\epsilon_j < \epsilon_i$, the time slot v is newly reserved for the packet p_i^k and the scheduler starts another search from the time slot v down to the current time slot in order to reserve another time slot for the packet $p_j^{k'}$. If the scheduler finds no time slot during the search, the packet p_i^k or $p_j^{k'}$ is inserted into the queue Q. At each time t, the packet that is reserved the earliest time slot in R is scheduled.

Since this algorithm takes into consideration the deadlines and the degradation values of the flows simultaneously, it is expected to perform well in the two aspects of our scheduling goal.

4.1.5 Example of Online Scheduling

We present a scheduling example of the algorithms that we consider. Assume there are four packets to be scheduled at time t as shown in Table 1. The schedules generated by the algorithms are shown in Table 2. For example, EDF would schedule p_1,p_2 or p_3 at time $t+1$. While EDF, EOG, and LFF schedule three packets in this example, GDF schedules only two packets. Packet p_2 associated with ϵ_{max} may not be scheduled by EDF, while it is scheduled by the other algorithms. While GDF schedules the most two lagging packets, LFF schedules the most three lagging packets.

Packet p_i	$d(p_i)$	ϵ_i
p_1	$t+2$	0.04
p_2	$t+2$	0.10
p_3	$t+2$	0.01
p_4	$t+3$	0.07

Table 1: Packets in Example

Algorithm	$t+1$	$t+2$	$t+3$
EDF	$p_1/p_2/p_3$	$p_1/p_2/p_3$	p_4
GDF	p_2	p_4	
EOG	p_2	p_1/p_3	p_4
LFF	p_1	p_2	p_4

Table 2: Schedules in Example

4.2 Wireless Error Handling

The presence of errors raises the issue of how to handle the situation where a packet was scheduled, but its delivery failed due to a channel error. This section presents a wireless error handling mechanism that can be used by any online scheduling algorithm.

In handling such a packet that experienced an error, one way is to drop the packet and the other way is to reschedule the packet at some time later. The packet drop would not be good if the packet can be rescheduled for a successful transmission later. Considering the bursty nature of wireless channel errors, rescheduling the packet immediately would not be good either. It is desirable to delaying rescheduling the packet to sometime later such that it can escape an error burst and can be delivered on time. Since errors are unpredictable, however, it is not clear how long rescheduling the packet should be delayed. We call this rescheduling delay *backoff time*.

The backoff time, b_i, of each flow f_i needs to be defined long enough to escape a potential bursty error and short enough for the packet to be scheduled prior to the expiration of its deadline. When a packet p_i^k experienced an error, we define b_i as follows:

$$b_i = (t + d(p_i^k))/2, \quad (3)$$

where t is the current time, and $d(p_i^k)$ is the deadline of a packet that experienced a failed delivery.

Initially $b_i = 0$, we assume that all flows have initially error-free channels. Then, b_i is used as an estimation of the status of channel ch_i. A packet of flow f_i is considered to be scheduled only if $b_i < t$, which means that its channel is predicted to be error-free at that time.

5 Simulation Results

This section evaluates the performance of the four algorithms based on simulations over the IEEE 802.11 PCF (Point Coordinator Function) protocol. Under the PCF protocol [12], the point coordinator (PC) that resides in the base station controls packet transfers using a polling scheme during contention free periods (CFPs). The IEEE 802.11 standard described a round-robin algorithm as a basic polling scheduling technique. We simulated the four algorithms as a polling scheduling algorithm instead of the round-robin algorithm respectively. The standard described each mobile host send a base station an association request message to get associated with the base station. This association request message contains a capability field including PCF related information such as whether the mobile host is able to response to polling and whether the mobile host wants to be polled during CFPs. Under the standard protocol, the capability field does not contain the real-time flow characteristics of the mobile host. However, we assume that the mobile host can pass its real-time flow characteristics to the PC through the capability field of the association request message, and then the PC can use the characteristics for scheduling. Hence, real-time scheduling algorithms such as the four algorithms can be used in the PCF protocol with one minor change to the standard protocol.

5.1 Simulation Environment

All the simulations were done in *ns*, which is a discrete event simulator developed by the VINT project at the University of California at Berkeley [10]. The CMU Monarch extensions [4] added wireless and mobility supports to *ns*, including a CSMA/CA medium access mechanism defined in IEEE 802.11. We added the 802.11 point coordination function (PCF) protocol to *ns*, in order to simulate scheduling algorithms over

the PCF protocol. We also added a feature to a MAC layer so that it drops a periodic packet when the deadline of the packet expires, which is not defined in the IEEE 802.11.

The capacity of the wireless medium is 11 Mbps following the IEEE 802.11b. The maximum CFP (contention-free period) duration is 49 msec, and the minimum CP (contention period) duration is 2 msec. Each simulation time is 60 seconds.

To model unpredictable location-dependent bursty errors, we used the following error model. For each channel ch_i, we created blackout period w_i^k where $1 \leq k \leq K$. Channel ch_i is in error state at any time instant during each blackout period w_i^k. Any packet delivery of flow f_i considered to be failed if the duration of the delivery is overlapped with any blackout period w_i^k. Since each flow f_i has its own blackout period w_i^k, this captures the location-dependent characteristics of the errors. The duration of each w_i^k is uniformly chosen in [2.5, 15] msec to represent bursty errors. The error duration rate is the ratio of $\sum_{k=1}^{K} w_i^k$ to the total simulation time. In our simulations, the error duration ratio ranges from 0.0 to 0.4 increasing by 0.05.

Simulations have been performed to evaluate the performances of EDF, GDF, EOG, and LFF scheduling algorithms. The number of flows used in our simulations and its corresponding offered load are shown in Table 3. For instance, the offered load of 6 flows corresponds to the 45% of the channel bandwidth. Each workload have four different flow combinations such that each combination has the same corresponding offered load. Assuming that each packet of f_i has the same length of 1460 bytes, we fill out the tuple (v_i, D_i, e_i) information of each flow f_i in order to roughly model a multimedia application requiring 512 kbps as follows: if $i \leq 6$, v_i is 20 msec; otherwise, v_i is 15 msec, if i is 4, 8, or 12, D_i is 15 msec; otherwise, D_i is 10 msec, and e_i is $i\times$ 0.01.

Number of Flows	6	9	12	15
Offered Load	45%	60%	95%	120%

Table 3: Workload in Simulation

5.2 Simulation Results

Figure 2 shows the performances of the four online algorithms in terms of maximizing the system throughput, T_{sys}, which is one aspect of our scheduling goal. We can see that EDF is still the best in maximizing the system throughput even with errors, among the four algorithms. The figure also shows that LFF provides almost the same system throughput as EDF. As expected, GDF is the worst in this performance measure.

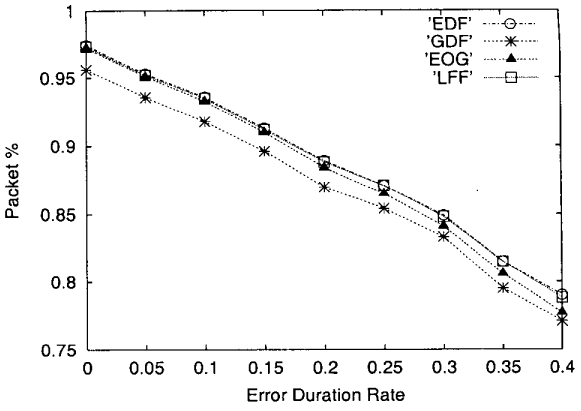

Figure 2: System Throughput ($\frac{\sum_{i=1}^{N} M_i^a}{\sum_{i=1}^{N} M_i}$)

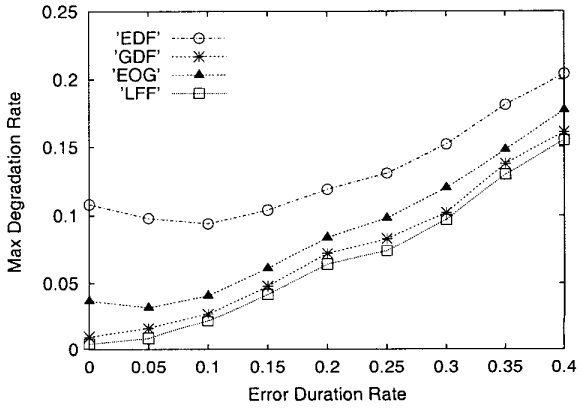

Figure 3: Maximum Degradation Value (ϵ_{max})

Figure 3 plots the performances of the algorithms in terms of minimizing the maximum degradation value among all flows, ϵ_{max}, which is the other aspect of our scheduling goal. Even though GDF is focused on minimizing ϵ_{max}, LFF performs better than GDF. We believe that this phenomenon can be explained by the fact that LFF achieves higher throughput than GDF. As one might expect, EDF performs poorly in comparison to all other algorithms.

In Figure 4, we measure the performances of the algorithms in terms of minimizing the maximum difference among the degradation values of all flows, $\max_{i,j} |\epsilon_i - \epsilon_j|$. This measure indicates how equally each scheduling algorithm minimizes ϵ_i for all flows. The figure shows that GDF provides the smallest value of $\max_{i,j} |\epsilon_i - \epsilon_j|$ among the four algorithms. This means that GDF tries to min-

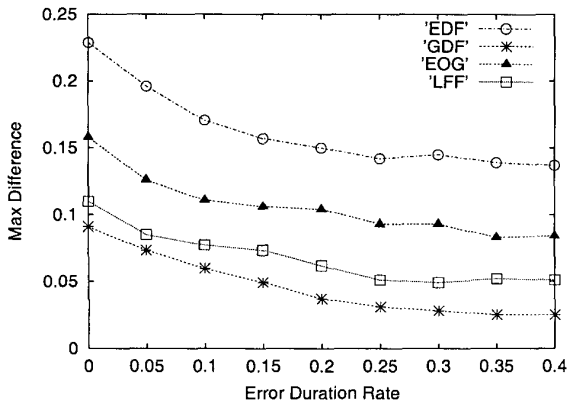

Figure 4: Maximum of $|\epsilon_i - \epsilon_j|$

imize ϵ_i the most equally among the four algorithms. While we capture fairness by minimizing ϵ_{max}, one may capture fairness in another way, for instance, by equally minimizing ϵ_i. From the latter viewpoint of fairness, GDF can be considered as a fairer scheduling algorithm than all the others. It is also shown that EDF performs poorly in another fairness measure.

6 Conclusion

We studied the problem of scheduling packets with deadlines in wireless network with unpredictable channel errors. We considered scheduling objectives of achieving fairness and maximizing the overall throughput. Fairness is to ensure that the maximum degree of degraded QoS among all real-time flows is minimized. We showed that no online algorithm can be guaranteed to achieve a bounded performance ratio for fairness objective. We then described and compared four online algorithms, namely, EDF, GDF, EOG, and LFF, using simulations. For the fairness objective, the algorithms in decreasing order of performance are LFF, GDF, EOG, and EDF. Whereas, for the maximum throughput objective, the algorithms in decreasing order of performance are EDF, LFF, EOG, and GDF. Thus LFF is the best of the four algorithms for simultaneously achieving both objectives. We also showed that there is a polynomial time offline algorithm that determines an optimal fair schedule with maximum throughput.

In this paper, we considered the case of scheduling the same length packets. Due to the non-preemptive nature of packet transmission, the problem of achieving the two aspects of our scheduling goal effectively in the presence of variable length packets is quite difficult, and we are presently studying the problem. We are also studying other measures of fairness such as max-min fairness.

References

[1] R.K.Ahuja, T.L. Magnanti, and J.B.Oslin, *Network Flows*, Prentice-Hall, Englewood Cliffs,1993

[2] H. Aydm, R. Melhem, D. Mosse, and P. Mejia-Alvarez, *Optimal Reward-Based Scheduling for Periodic Real-Time Tasks*, IEEE RTSS, 1999.

[3] S. K. Baruah, N. K. Cohen, C. G. Plaxton, and D. A. Varvel, *Propotionate Progress: A Notion of Fairness in Resource Allocation*, Algorithmica, Vol. 15, No. 6, pp. 600-625, June 1996.

[4] *The CMU Monarch Project's Wireless and Mobility Extensions to ns*, The CMU Monarch Project. August 1999. Available from http://www.monarch.cs.cmu.edu/.

[5] J. Capone and I. Stavrakakis, *Delivering Diverse Delay/Dropping QoS Requirements in a TDMA Environment*, ACM/IEEE MOBICOM Conference, September 1997.

[6] C. Chang, K. Chen, M. You, and J. Chang, *Guaranteed Quality-of-Service Wireless Access to ATM Networks*, IEEE Journal of Selected Areas in Communications, Vol. 15, No. 1, January 1997.

[7] J.-Y. Chung, J. W.-S. Liu, and K.-J. Lin, *Scheduling periodic jobs that allow imprecise results*, IEEE Transactions on Computers, Vol. 19, No. 9, 1156-1173, September 1990.

[8] S. Choi and K. G. Shin, *A Unified Wireless LAN Architecture for Real-time and Non-real-time Communication Services*, ACM Transaction on Networking, Febrary 2000.

[9] J. K. Dey, J. Kurose, and D. Towsley, *On-line Scheduling Policies for a class of IRIS (Increasing Reward with Increasing Service) Real-Time Tasks*, IEEE Transactions on Computer, Vol. 45, No. 7. 802-813, July 1996.

[10] Kevin Fall and Kannan Varadhan, editors. *ns Notes and Documentation*, The VINT Project, UC Berkeley, LBL, USC/ISI, and Xerox PARC, November 1997. Available from http://www-mash.cs.berkeley.edu/ns/.

[11] M. Hamdaoui and P. Ramanathan, *A Dynamic Priority Assignment Technique for Streams with (m,k)-firm Deadlines*, IEEE Transactions on Computers, Dec. 1995.

[12] IEEE, *IEEE Std 802.11 - Wireless LAN Medium Access Control (MAC) and Physical layer (PHY) Specifications*, 1997.

[13] IEEE, *IEEE Std 802.11a - Wireless LAN Medium Access Control (MAC) and Physical layer (PHY) Specifications: High Speed Physical Layer(PHY) in the 5 GHZz Band*, 1999.

[14] IEEE, *IEEE Std 802.11b - Wireless LAN Medium Access Control (MAC) and Physical layer (PHY) Specifications: High Speed Physical Layer(PHY) in the 2.4 GHZz Band*, 1999.

[15] S. Lu, V. Bharghavan, and R. Srikant, *Fair Scheduling in Wireless Packet Networks*, IEEE/ACM Transactions on Networking, Vol. 7, No. 4, August 1999.

[16] S. Lu, T. Nandagopal, and V. Bharghavan, *A Wireless Fair Service Algorithm For Packet Celluar Networks*, ACM/IEEE MOBICOM Conference, October 1998.

[17] T. Ng, I. Stoica, and H. Zhang, *Packet Fair Queueing Algorithms for Wireless Networks with Location-Dependent Error*, ACM/IEEE MOBICOM Conference, October 1998.

[18] W.-K. Shih and J. W.-S. Liu, *On-line Scheduling of Imprecise Computations to Minimize Error*, IEEE Real-Time Systems Symposium, December 1992.

[19] D.D. Sleator and R.E.Tarjan, *Amortized efficiency of list update and paging rules* Communications of the ACM, 28(2): 202-208, 1985.

[20] R. West, K. Schwan, and C. Poellabauer, *Scalable Scheduling Support for Loss and Delay Constrained Media Streams*, IEEE Real-Time Technology and Application Symposium, 1998.