Scheduling Real-time Traffic With Deadlines over a Wireless Channel *

Sanjay Shakkottai [a] R. Srikant [b]

[a] *Coordinated Science Laboratory and*
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
shakkott@uiuc.edu
[b] *Coordinated Science Laboratory and*
Department of General Engineering
University of Illinois at Urbana-Champaign
rsrikant@uiuc.edu

Recently, there has been widespread interest in the extension of data networks to the wireless domain. However, scheduling results from the wireline domain do not carry over to wireless systems because wireless channels have unique characteristics not found in wireline channels, namely, limited bandwidth, bursty channel errors and location-dependent channel errors.

In this paper, we study the problem of scheduling multiple real-time streams with deadlines, over a shared channel. We show that, in general, unlike the wireline case, the earliest due date (EDD) or shortest time to extinction (STE) policy is not always the optimal policy, even if the channel state is perfectly known and EDD is implemented only over channels in a "Good" state. Here, optimality is measured with respect to the number of packets lost due to deadline expiry. However, for most values of the channel parameters that are of practical interest, we show through analytical and numerical results that the EDD policy over "Good" channels is nearly optimal. Finally, through simulations, we also show that by combining this policy with fair scheduling mechanisms would result in scheduling algorithms that provide some degree of isolation between the sources as well as provide a natural way of compensating channels that see prolonged error bursts.

* Research supported by NSF Grants ANI-9714685 and NCR-9701525

1. Introduction

The anticipated emergence of mobile integrated-services networks depend on the ability of the network to arbitrate among various data sources requiring different quality-of-service (QoS) requirements over shared wireless links. Typically, the traffic is expected to be a mix of real-time traffic such as multimedia teleconferencing and voice, and data-traffic such as WWW browsing and file transfers.

The problem involves suitably allocating resources the contending sessions, depending on each session's desired QoS. The problem can be studied in two parts, namely, *admission control* and *packet scheduling*. Admission control deals with the amount of traffic (number of streams) that is allowed to enter the network, subject to the constraint that the desired QoS for various users are met. On the other hand, the scheduling problem deals with allocating the bandwidth (the shared resource) among the various sources that have already been admitted on a packet-by-packet basis. In this paper, we study only the scheduling problem, i.e., given some traffic with QoS constraints, we would like to allocate the bandwidth on a packet-by-packet basis such that the desired QoS is met.

1.1. Related Work for Wireline (Error-Free) Channels

The scheduling problem for wireline systems has been studied extensively. One class of traffic that has been studied is real-time traffic, which is modeled as a stream of packets, with each packet having an expiry time beyond which the packet is of no use to the end user. The objective of the scheduler is to transmit each packet before its expiry, and if this not possible, to minimize the number of lost packets due to deadline expiry. It has been shown in [17] that the shortest time to extinction (STE) policy, also known as earlier due date (EDD) or earlier deadline first (EDF), is optimal for such traffic. Further, [3] has shown that for a single queue having packets with deadlines and the service distribution being exponentially distributed, the STE policy is optimal. In [12], the authors make the observation that, if losses do occur with the EDD policy, then EDD is not the only optimal policy. Schedulability conditions are derived in [7] for preemptive and non-preemptive deadline-ordered service disciplines for a system with a constant capacity. The authors in [9] study a wireline system consisting on N queues and a single server. The server arbitrates among packets of different lengths, each with a deadline. The arrival process to each queue is a continuous

time (σ, ρ) (see [4]) constrained process. A set of arrival processes is said to be schedulable under a policy π if every packet is scheduled before its deadline expiry. They show that among the class of all non-preemptive policies, the policy which results in the largest schedulable region is a variant of the earliest due date policy.

Another aspect of scheduling that has attracted a lot of research attention is fairness. A popular model for fair resource allocation to fluid sources over a link is the Generalized Processor Sharing (GPS) model (see[5]). In [18] and [19], the authors have studied the performance of a network of queues, each queue having a GPS server. They have analyzed the worst case delay of packets when the sources are constrained by leaky buckets. There are several packet-level algorithmic implementations (see [10,11]) which try to approximate fluid GPS as closely as possible. In GPS, when a flow has nothing to transmit during some time, it is not allowed to reclaim the channel capacity that would have been allocated to it in that interval. There are other policies such as that in [22] where the session can reclaim the capacity.

In [6], the authors note that the objective of recent research in fair queuing schemes has been to emulate a GPS system as closely as possible; the motivation being that, by using this scheme, we can guarantee worst case end-to-end delay and guaranteed bandwidth. A consequence of such systems is that when some sessions are not back-logged, any excess bandwidth is distributed among the back-logged connections in proportion to their weights. However, weights are set based on long term needs and not in a state-dependent manner based on instantaneous needs. The authors in [6] propose a few modified fair scheduling schemes which preserve the rate guarantees of GPS but distribute the excess bandwidth adaptively is a state dependent manner such that other constraints such as losses or delays are reduced.

1.2. Wireless Channels

All the previous work that have we described so far pertain to the wireline case. The question is whether these results carry over in some simple manner to the wireless domain. Before we discuss this, we note some peculiarities found in wireless scheduling, but not in wireline scheduling. First, the channel here is not perfect and is subject to errors. This causes causes *bursts of errors* to occur during which packets cannot be successfully transmitted on link. In conventional

wireline resource allocation problems, it is assumed that the channel is perfect, hence the scheduling policy need not consider the channel state for making a decision. Secondly, the channel *errors are location-dependent*, i.e., a particular frequency (channel) may be accessible to some users but not to others since the users are typically mobile stations with possibly different fading characteristics. This means we may not be able to successfully schedule a users' packet on the link even though fairness may dictate this, but we may be able to schedule another user's packet. Because of these two reasons, scheduling policies which are good in the wireline case are not necessarily the best policies in the wireless domain.

In [21], optimal scheduling for a wireless system consisting on N queues and a single server is studied. The arrival process to each queue are assumed to be and i.i.d. Bernoulli processes. The channel perceived by each queue is also assumed to be an i.i.d. Bernoulli process. The authors show that the policy which minimizes the total number of packets in the system in a stochastic ordering sense is the one which serves the longest queue. We note that in this study, the authors did not have any deadlines for the packets. For recent related work on non real-time traffic, see [14].

As in the wireline case, fair queueing has been studied for the wireless scenario. The authors in [15,16] propose a new model for wireless fair scheduling based on an adaptation of GPS to handle location dependent error bursts. Given the arrival process for each of the flows and the error patterns perceived by each of the flows, the authors define an *error-free service* as the GPS service for the flows with identical arrival processes and completely error-free channels. A flow is said to be lagging if, at any time instant, its queue length is longer than that of the error-free counterpart at the same instant. Similarly, a flow is said to be leading if its queue length is shorter than that of its error-free counterpart. The key feature of their fluid fair queuing model is to allow lagging flows to make up lag by causing leading flows to give up their lead. By bounding the amount of lag and lead, they show that they can tradeoff between long-term fairness and separation between flows.

The authors in [1] study the effect of the wireless link on the performance of transport protocols such as TCP for various scheduling protocols using a simulation-based approach. The conclude that a channel-state dependent scheduler can lead to significant improvement in channel utilization for typical wireless LAN configurations. This idea is further explored in [8] in the context of fair queueing.

However, no study of optimal policies for real-time traffic with deadlines over wireless systems seems to exist in current literature.

1.3. Our Contribution

In this paper, we study a modified version of EDD, which we call the *feasible earlier due date* (FEDD) policy. This is basically a channel state-dependent EDD policy where the scheduler chooses to schedule the packet which has the earliest time to expiry from among the set of queues whose channels are "Good". A policy of this type was shown to be optimal in the wireline case (for the case of i.i.d Bernoulli arrival and channel processes) as noted earlier. In contrast, for the wireless case, [21] has noted that a queue length based policy is optimal for minimizing overall delay in the system for a system without deadlines. Unlike this, in our case, with Markovian channels and strict delay constraints, one would expect that both deadlines and delays should figure in the optimal policy. In this paper, we show that this is indeed the case. However, *we will see that just looking at the deadlines is sufficient in most cases.* In fact, we will show that for certain restricted arrival processes FEDD is the delay-optimal policy. Further, we quantify the magnitudes of the tradeoff between queue lengths and deadlines. We construct "worst-case" systems for the FEDD policy, and even in this case, we will see that we really don't need to look at queue lengths for most cases. The usefulness of this result lies in the fact that this allows us to implement simple scheduling policies which perform well most of the time. We note that, to obtain the optimal policy, we would have to solve a very complex dynamic program taking into account the channel and arrival process parameters at each time to schedule a packet. This is clearly not feasible.

We realize that the FEDD policy can by itself be unfair, but proper traffic policing can lead to fairness. One may object that traffic policing may result in the overall system not being work-conserving. To partially overcome this problem, one may use a rate-proportional scheduler which provides guaranteed minimum bandwidth to each connection and distributes the residual bandwidth using the the FEDD criterion in a manner analogous to the state-dependent excess-bandwidth sharing as proposed in [6]. This would combine the two aspects of scheduling, namely, fairness and optimal scheduling in a single-framework. The results in this paper provide a stepping stone to achieve this end.

2. Description of Model

Consider a discrete-time queuing system consisting of N queues, such that each queue is associated with a channel which, at any time t, can be in one of two states: "Good" or "Bad". This models the physical scenario where a particular user could be in a fade (corresponding to the "Bad" state). Hence, if the channel of queue i is in the "Bad" state at time t, and a packet from queue i is transmitted on the link, this results in packet loss and the packet has to be retransmitted later. We assume that the channel state is modulated by a two-state discrete-time Markov chain also known as the Gilbert-Elliot model. By suitably choosing the transition probabilities of the chain, we can model different fading characteristics. We denote the transition probabilities by P_{gb} and P_{bg}, where

$P_{gb} \triangleq$ The transition probability from the "Good" state (G) to the "Bad" state (B) of the Markov chain modulating the channel.

$P_{bg} \triangleq$ The transition probability from the "Bad" state (B) to the "Good" state (G) of the Markov chain modulating the channel.

The scheduler can schedule only one packet per slot. In the next few sections, we assume that the scheduler has perfect knowledge of the current states of the channels. We shall later study the case where channel state is estimated. Since the channel state is known, the scheduler chooses a packet from among the set of packets whose channels at t are in the "Good" state. Associated with each packet is a deadline beyond which the packet cannot be scheduled. We assume that the packet deadlines in a queue are ordered in a non-decreasing manner according to their position in the queue.

We measure the performance of the scheduling policy in terms of the expected number of packets lost at the node due to packet deadline expiry over any finite interval of time. Unless otherwise stated in any result, we term the policy which minimizes this quantity as the optimal policy. This measure, we realize, does not take into account the fairness of the policy. However, as discussed earlier, fairness can be achieved by proper policing policies. Later, we provide simulation results for such a policy.

3. Properties of the System

In this section, we describe the FEDD policy. We then show some properties of the queuing system when this policy is used for scheduling.

The *Feasible Earliest Due Date* (FEDD) policy is a scheduling policy which at each time t schedules that packet which has the smallest time to expiry and whose channel at time t is in the "Good" state. In the wireline case where the channels are assumed to be perfect, it has been shown (see [17]) that this is the optimal policy (for every sample path of the arrival process). This is not true if the channels have errors. However, in this section, we show that this policy is optimal even with channels which have errors in some special cases. Before we can do so, we need the following results, whose proofs are straightforward.

Lemma 3.1. Consider a static queuing system consisting of N queues; each queue i having only one packet with deadline d_i. The channel process for the queues are i.i.d. Markovian as described above. Then, at each time t, the optimal scheduling policy is FEDD. ∎

Lemma 3.2. Consider a single queue having many packets with deadlines. As before, let the deadlines of packets be ordered in a non decreasing manner depending on the packets' position in the queue. Let the channel process be two state, but be modulated by some general process. Then, the optimal scheduling policy which minimizes the number of packets lost over every sample path is to schedule the Head of Line (HOL) packet. ∎

Let us now consider a more general scenario where FEDD is optimal. Consider a system consisting of many queues, the channels corresponding to the queues being independent and identical Markov channels. Let the queues be initially empty. Let the arrival processes to the queues be identical deterministic periodic processes (though possibly phase-shifted), i.e., a new packet arrives every T slots in each queue, but the packets need not arrive at the same slot in each queue. A packet's deadline is $d + t_{arrival}$, where $t_{arrival}$ is the packet's arrival epoch.

Proposition 3.1. For the above system, FEDD is the optimal scheduling policy.

Proof: We only provide the proof for the two-queue case, the general proof is very similar. Suppose that there are l packets in queue 1 and m packets in queue 2 time t to the two queues. Denote the deadline of the HOL packet in queue i by d_i. We make the following observations:

(i) $d_1 = d_2 \Rightarrow \begin{cases} l = m \\ \text{The arrival processes are in phase} \end{cases}$

(ii) $d_1 < d_2 \Rightarrow l \geq m$. Further, if $d_1 < d_2$ and the processes are in phase, i.e., arrivals occur in the same slot to both queues, then $l > m$.

In Case (i), as the two queues are in the identical state and future arrivals are the same, it is immaterial which HOL packet we schedule. Hence, FEDD is trivially the optimal policy.

Now consider Case (ii). If the arrivals are in phase, then $l > m$. Now, since the arrivals are in phase, some packet in queue 1 will have a deadline equal to that of the HOL packet of queue 2. Further, every packet in queue 2 will have a corresponding packet in queue 1 with the same deadline. Suppose we claim that scheduling the HOL packet from queue 2 is optimal. Then, as the channels and future arrivals are identical to both queues, the expected number of packets lost would be identical if we schedule the packet from queue 1 with deadline d_2. Thus, the original scheduling decision can be thought as a decision between scheduling the packet with deadline d_2 in queue 1 and the HOL packet in queue 1. However, by scheduling the packet from queue 1 with deadline $d_1 < d_2$ (i.e., the HOL packet of queue 1), it follows from Lemma 3.2 that the performance will be better. Hence, FEDD is the optimal policy in this case.

If the arrivals are not in phase, we can use a similar coupling argument as in the "in-phase" case to show that FEDD is optimal. ∎

The above result is true for any identical, in-phase deterministic arrival process (not necessarily periodic) to both the queues. The proof is analogous to the one above. The next question to ask is whether FEDD is optimal for some general arrival processes to the queues. As observed earlier, FEDD need not be the optimal policy in this case. The optimal policy depends on the deadlines of all the packets in the queues and future arrivals. To illustrate this, consider the following example:

Example 3.1. Consider a two queue system with i.i.d. Markovian channels. Let queue 1 have 1 packet with deadline 10, i.e., the packet expires in 10 time units from the current time. Let queue 2 have 3 packets with deadlines 12, 14 and 16 respectively. Suppose that from the next time slot, back to back packets arrive to queue 1, then it is clear that the optimal policy now would be to schedule the packet from queue 1. On the other hand, if a burst arrives to queue 2, then the current optimal policy would be to schedule packets from queue 2. We can see that if arrival occur to both queues with different deadlines, the current optimal policy is going to be a function of future arrivals and all their deadlines. ■

We saw in the example the FEDD need not be the optimal scheduling policy. In practice, there will be a tradeoff between the packet deadlines and (current and future) queue lengths. Hence, the general optimal policy would be a combination of a due-date based one and a queue-length based one.

We could solve this problem by a dynamic program formulation. Then, given an arrival process with some sequence of deadlines to each queue, we could find that policy which minimizes the expected number of packets dropped per unit time. However, in a realistic scenario, the queues are distributed among various users in the network and the scheduler is at the base station. It seems difficult for the scheduler to acquire information about the state and deadlines of all packets in every queue for each time-slot as such a scheme would lead to loss of bandwidth. Instead, we consider a polling model as shown in Figure 1. Here, the scheduler at the base-station polls various queues during the polling period and schedules all the packets during the packet transmission phase. There are several MAC layer issues associated with this which we shall not go into. For details, see [15] and [2]. In the following section, we derive the optimal scheduling policy for this model. We will later show that even in the more general scheduling problem where the state information is continuously available to the scheduler, simulations indicate that the optimal policy for the polling model seems to carry over to the general scheduling model.

4. Optimal Scheduling with the Polling Model

In this section, we study the optimal scheduling policy for the polling model. We can clearly see that in this model, the scheduling problem reduces to a se-

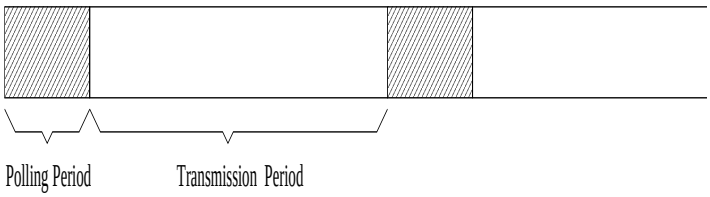

Figure 1. Polling model: The scheduler acquires data about the state of various queues during the polling period and suitably schedules the data during the transmission period.

quence of static scheduling problems, i.e., we have some initial configuration of the queues. We assume that no new arrivals occur to any of the queues. Packets are scheduled till either all packets are transmitted or expire. The poll-transmit cycle then repeats. Hence, in the following sections, we derive the optimal static schedule.

4.1. Analysis of the Static Queuing Model

As in the earlier sections, consider first a two-queue system as shown in Figure 2. Initially, let there be $l \leq L$ packets in queue 1 and only 1 packet in queue 2. Let the deadline of the HOL packet in queue 2 be $d_2 < d_1 \leq d$. Thus, we have two queues, queue 2 having only one packet but with the smallest deadline and queue 1 having many packets but with deadlines larger than that of the packet in queue 2. By making the number of packets in queue 1 large, we would make it less likely for FEDD to be optimal, and this would allow us to study the trade-off between queue lengths and deadlines. Let the deadlines of successive packets in queues 1 be at least $t_a \geq 1$ apart. This corresponds to a peak rate constraint for packets arriving to a queue. We impose this constraint as we would like to understand how dynamic systems will behave from the results of this static scheduling problem. The channel processes, as before are i.i.d. Markov processes with transition probabilities as before. In this section, we find regions of $(l, d_1, d_2, t_a, P_{gb}, P_{bg})$ where FEDD is an optimal scheduling policy.

Even in the this static set-up, in general, to determine whether FEDD is optimal for a particular pair of P_{gb} and P_{bg}, we have to solve a dynamic program for every possible $l \leq L$ and $d_1, d_2 \leq d$ with the constraint that $d_2 < d_1$. However, it turns out that we do not have to study every case. We will show in in Proposition 4.1 that if FEDD is optimal for $l = L$ and $d_1 = d, d_2 = d - 1$, then FEDD is optimal for all $l \leq L$ and $d_1, d_2 \leq d$. Then, we will numerically

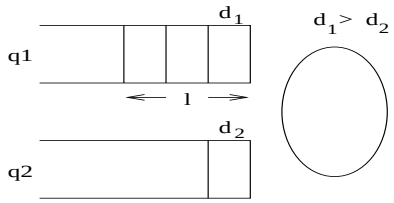

Figure 2. To study the tradeoff between queue length and deadlines, we consider the model where queue 2 has only 1 packet but with a deadline earlier than that of the HOL packet of queue 1.

determine if FEDD is optimal for $l = L$ and $d_1 = d, d_2 = d - 1$. If it is optimal for this case, then from Proposition 4.1, it is optimal for all $l \leq L$ and $d_1, d_2 \leq d$.

Proposition 4.1. FEDD is an optimal scheduling policy under the above constraints for a given P_{gb} and P_{bg}, if FEDD is the optimal scheduling policy for the following system:

> The deadline of the HOL packet in queue 2 is $d - 1$ and that of queue 1 is d. The packet deadlines in queue 1 are separated by exactly t_a and there are exactly L packets in queue 1.

Proof: See Appendix A. ∎

In view of the above proposition, we consider the case where $l = L$ with $d_1 = d$ and $d_2 = d - 1$. Then, it follows that the deadline of the last packet in queue 1 is $D_{max} = d + (L - 1)t_a$. After time D_{max}, the deadline of the last packet in queue 1, no packets are left in the system. Hence, the optimal policy for the resulting system can be solved iteratively going backward in time from D_{max} using a dynamic program. This still results in a problem of exponential complexity in D_{max}. It turns out that we can further reduce the complexity by noting the following corollaries of Proposition 4.1. These results are an immediate consequence of the proof of this proposition in Appendix A.

Let the quadruple vector $(L, 1, d, d-1)$ denote the state of the queues. Here the first and second elements represent the number of packets in queues 1 and 2 respectively and the third and fourth elements represent the deadlines of the HOL packets in queues 1 and 2 respectively.

Corollary 4.1. Suppose the initial state of the system is $(K, 1, d, d - 1)$ where

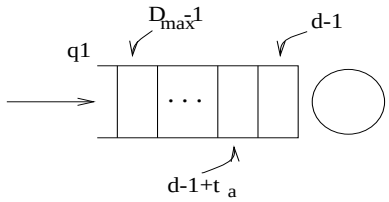

Figure 3. The two-queue system at $t = 1$ degenerates into a single queue.

the first and second elements represent the number of packets in queues 1 and 2 respectively and the third and fourth elements represent the deadlines of the HOL packets in queues 1 and 2 respectively. Suppose that FEDD is the optimal scheduling policy for this system. Then, the optimal policy for all initial states $(K - r, 1, d - x, d - x - 1), r \geq 0, x \geq 0$ is again FEDD. ∎

Corollary 4.2. At time t with the initial state being $(K, 1, d, d-1)$, if FEDD results in fewer expected packet lost than $\mathcal{F}$, the policy which schedules the packet from queue 1 initially but follows the FEDD policy for all future times, FEDD results in fewer expected packets lost than all other causal policies. ∎

From Corollary 4.2, we have to compare FEDD only with the policy $\mathcal{F}$ to determine whether FEDD is indeed optimal. Consider the queuing system with initial state $(K, 1, d, d - 1)$. Suppose that both the channels (corresponding to the 2 queues) are in the "Good" state. Then, the scheduler can either decide to schedule the packet from the shorter queue (with the HOL packet having an earlier deadline) or the packet from the longer queue (with the HOL packet having a later deadline). Irrespective of the policy used, in the future, FEDD is used. If we show that FEDD (i.e., schedule the packet with the earlier deadline) leads to a smaller expected number of packets lost, then, it follows from Proposition 4.2 that for the initial state $(K, 1, d, d - 1)$, FEDD is the optimal policy.

To perform this computation, we compare the expected number number of packets lost under two scheduling policies. In one case, FEDD is used for all time while in the other case, we schedule the packet from the other queue (which has larger deadline and longer queue length) initially, but use FEDD for all future times.

In the first case, we schedule the packet with the earlier deadline which

results in only one active queue being present from the next (discrete) time epoch. Then, the 2 queue system degenerates into a single queue system as shown in Figure 3.

Let us define $E_l(t,k,S)$ as the expected number of packets lost from time $t, 0 \leq t \leq D_{max}$ with k packets in queue 1 and the state of the channel being $S \in \{\mathrm{G},\mathrm{B}\}$, where G and B correspond to the channel state at time t being "Good" or "Bad".

Then, the expected number of lost packets can be evaluated using the following recursive equations:

If $k \geq 1$,

$$E_l(t,k,B) = \begin{cases} P_{bg}E_l(t+1,k,G) + P_{bb}E_l(t+1,k,B) & \text{if } \tilde{d} \geq 2 \\ 1 + P_{bg}E_l(t+1,k-1,G) + P_{bb}E_l(t+1,k-1,B) & \text{if } \tilde{d} = 1 \end{cases} \tag{4.1}$$

$$E_l(t,k,G) = P_{gg}E_l(t+1,k-1,G) + P_{gb}E_l(t+1,k-1,B) \tag{4.2}$$

where $\tilde{d} \triangleq (D_{max} - t - (k-1)t_a)$ is the deadline of the HOL packet at time t. The recursion terminates at $k = 0$. If initially, there are K packets in queue 1, then, the expected number of packets lost is given by

$$E_{FEDD}(\text{lost}) = P_{gg}E_l(1,K,G) + P_{gb}E_l(1,K,B) \tag{4.3}$$

Now, we consider the other case where we initially schedule the packet from the longer queue. However, FEDD is used for all future times. Then, as in the earlier case, the expected number of packets lost can be evaluated using a set of recursive equations.

Let us define $\mathcal{E}_l(t,k,l,S_1,S_2)$ as the expected number of packets lost from time $t, 0 \leq t \leq D_{max}$ with k packets in queue 1, l packets in queue 2 and the state of the channels of the two queues being $S_1, S_2 \in \{\mathrm{G},\mathrm{B}\}$, where G and B correspond to the channel state at time t being "Good" or "Bad". Further, we define d_1 and d_2 as the times to expiry for the HOL packets of queues 1 and 2 respectively (given that k and l are ≥ 1). Then, at time t,

$$d_1 = D_{max} - t - (k-1)t_a \tag{4.4}$$

$$d_2 = (D_{max} - t - (K-1)t_a - 1) \tag{4.5}$$

With these definitions, we can now write the recursive equations to compute the expected number of packets lost. The equations are given in Appendix B.

As in the earlier case, the recursion terminates at $k = 0$. If initially, there are K packets in queue 1, then, the expected number of packets lost is given by

$$\begin{aligned} E_{\mathcal{F}}(\text{lost}) = P_{gg}^2 \mathcal{E}_l(1, K, 1, G, G) + P_{gg} P_{gb} \mathcal{E}_l(1, K, 1, G, B) \\ + P_{gb} P_{gg} \mathcal{E}_l(1, K, 1, B, G) + P_{gb}^2 \mathcal{E}_l(1, K, 1, B, B) \end{aligned} \tag{4.6}$$

Hence, by comparing the results from Equations (4.3) and (4.6), we can determine the ranges of P_{gb}, P_{bg}, d, t_a and K for which FEDD is the optimal scheduling policy. It can be shown that if FEDD is optimal for two queues, the it is the optimal scheduling policy for $N \geq 2$ queues under the condition that the same overall load is maintained in the new queuing system.

Remark 4.1. We note that we have a method of finding the regions where FEDD is optimal when the channel state is assumed to be known for the polling model. Simulations results presented later indicate that the results hold in a general dynamic case where the channel state is **estimated**. The assumption of perfect knowledge of the channel state is made in most analyses in literature [1,15,16] as it has been known that one-step prediction is a good approximation and which can be implemented in the MAC layer. For details, see [15]. ∎

5. Numerical Results and Simulation

5.1. Queue Length versus Deadlines for the Static Problem

In this section, we numerically compute values of P_{gb}, P_{bg}, d, t_a and K for which FEDD is the optimal scheduling policy. Due to space limitations, the numerical results primarily concentrate on the two-queue problem here. However, we present simulation results in the dynamic case for a 10 queue system. We show that the difference in queue lengths need to be quite large before FEDD ceases to be optimal.

The figures in this section all plot the normalized difference in the expected number of packets lost from Equations (4.3) and (4.6) versus P_{bg} for a fixed P_{gb} for various values of queue lengths. We refer to the difference as the *gain* due to FEDD. In the figures, we plot the regions where the gain is greater than zero. For a particular set of parameters where the gain is positive, from the results in the previous section, FEDD is the optimal scheduling policy for those parameters.

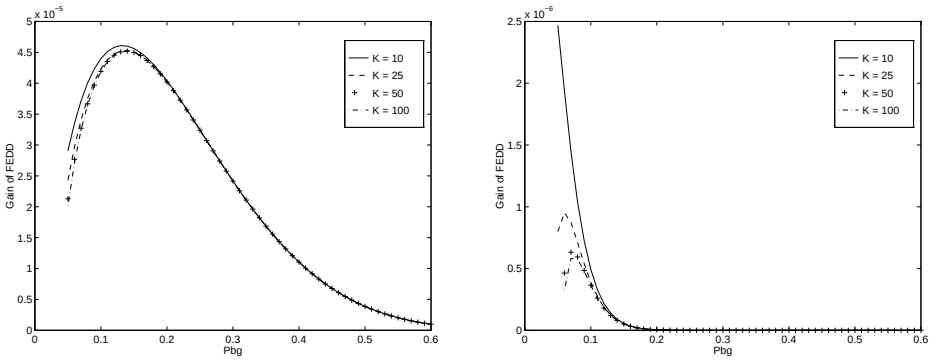

Figure 4. Numerical results: Absolute value of the normalized gain of FEDD vs. P_{bg} for $P_{gb} = 0.001$, $t_a = 2$, $d = 10$ (figure on the left) and $t_a = 2$ and $d = 50$ for various values of K. FEDD is the optimal policy for all values in this range. In the case of $d = 50$, we remark that FEDD is not optimal only for $K = 100$ and $P_{bg} = 0.05$.

As we have a discrete-time model for the system, we can use the results presented here for various actual systems. For example, if we consider packets of 200 bytes each with the link speed being 1.6 Mbps, then a packet transmission time is 1 msec. For a Raleigh fading channel of mean fade duration being 10 msec and the channel being "Good" for 99% of the time, $P_{gb} = 0.001$ and $P_{bg} = 0.1$ [13]. On the other hand, if we consider a wireless ATM network with 53 byte packets with 1.6 Mbps link, then a packet transmission time is 265 μsec. Then, for a fast fading channel with mean fade duration being 1 msec with fraction of "Good" time being 99.5%, $P_{gb} = 0.001$ and $P_{bg} \approx 0.25$. We can similarly interpret these graphs for various systems.

As in the earlier sections, the system has K packets in queue 1 and only 1 packet in queue 2. We assume that both channels are initially in the "Good" state.

Figure 4 plots the gain of FEDD versus the channel transition probability P_{gb} when $t_a = 2$ and $d = 10$. The small value of d means that the HOL packet from queue 2 (having just 1 packet) expires quite soon. $t_a = 2$ means that a packet expires only every alternate time slot which corresponds to a lightly loaded system. Intuitively, we would expect this to be the "best case" for FEDD as we weight in favor of earliest due date by having a packet expire very soon and by not heavily penalizing the policy disregarding the queue length by giving enough time between expiry of packets. We can see from the figure that the gain is positive for all values of P_{bg} irrespective of the number of packets in queue 1.

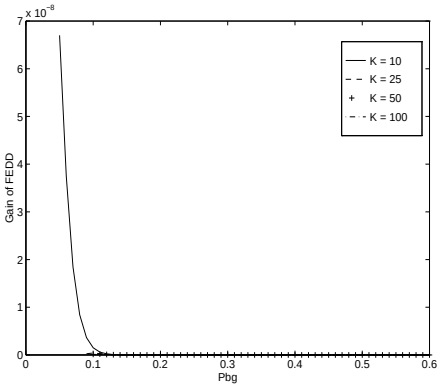

Figure 5. Numerical results: Absolute value of the normalized gain of FEDD vs. P_{bg} for $P_{gb} = 0.001$, $t_a = 2$ and $d = 100$ for various K. Here, FEDD is not optimal in the range $P_{bg} < 0.09$. However, we note that in the region where FEDD is optimal, the gain is orders of magnitude larger than the greatest loss.

In Figure 4, we increase the earliest deadline to $d = 50$ keeping t_a the same as in the previous plot. We again see that except for very small P_{bg} (corresponding to very long "bad" periods), FEDD is the optimal policy for even a very large difference in the queue lengths (i.e., $K = 100$).

Finally, in Figure 5, we increase the earliest deadline to $d = 100$. Even in this case, for $P_{bg} > 0.09$ and even large K, we see that FEDD is optimal. For small K, FEDD is optimal for almost the entire range of P_{bg} under study. From this series of plots, we can see that for lightly loaded systems, FEDD seems to be a reasonable policy to adopt for packet scheduling.

In the next two plots, we increase t_a to 1 and repeat the earlier study. This corresponds to a system with heavy load. In fact, as $K \to \infty$, the normalized load $\to 1$. In Figure 6, we keep $d = 10$ and as before study the performance as a parameter of P_{bg} for various values of K. We can see from the figure that for K of 10 and 50, FEDD is optimal for practically the entire range of P_{bg} under study. For $K = 100$, for $P_{bg} > 0.14$, FEDD is optimal. From our earlier discussion of the correspondence of these figures to real systems, we see that FEDD is the optimal scheduling for realistic channels for even the case where just one packet is present in one of the queues while 100 packets are present in the other but has a marginally higher deadline.

Finally, we keep the same load (i.e., $t_a = 1$) but we increase the deadline for the HOL packet to 50. This is in some sense, the worst case for the FEDD

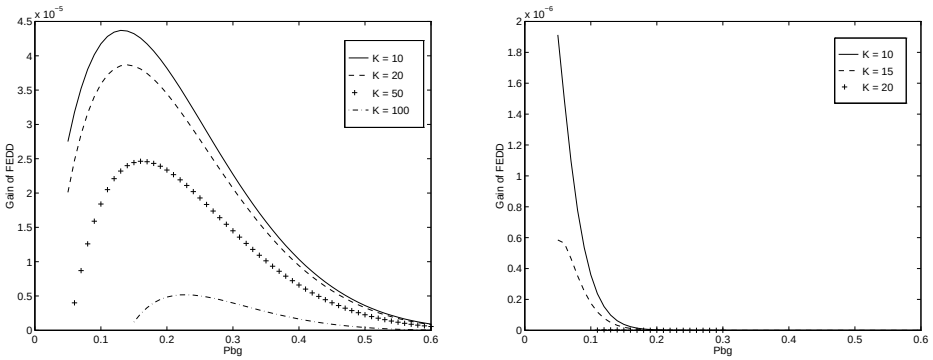

Figure 6. Absolute value of the normalized gain of FEDD vs. P_{bg} for $P_{gb} = 0.001$, $t_a = 1$, $d = 10$ and $t_a = 1$, $d = 50$ for various K. These correspond to a much more highly loaded system as compared to the earlier plots. We note that in case of $d = 10$ (plot on the top), FEDD is (possibly) not optimal for $K = 100$ and $P_{bg} < 0.14$. In the case where $d = 50$, FEDD is optimal for the interval $[0.1, 0.3]$ for $K = 20$ and for lower values of K, it is optimal over the whole interval we plot for.

policy. There is very small advantage due to the earlier deadline, but the queue length difference is very large. We see from the figure that even in this case, for $K = 10$, FEDD is the optimal scheduling policy. However, for $K = 20$, FEDD ceases to be the optimal scheduling policy for all values of P_{bg}. However, even for this extreme case, we can see from Figure 6 that for values of P_{bg} between 0.1 and 0.3 (this range corresponds to a reasonable model for the channel), FEDD is still optimal.

5.2. FEDD with Bursty Arrival Processes and Perfect Knowledge of the Channel State

We have so far studied the polling model and have found regimes where FEDD is optimal. We now look at the general scheduling problem where the state information is assumed to be known to the scheduler at all instants of time. We conjecture that the scenarios studied in the polling model represent "extreme" cases which could arise in the general dynamic problem. For approximately similar loads to various queues and identical channel processes with relatively short "Bad" times, it seems reasonable to assume that extreme imbalances in queue lengths will not occur. Hence, in some sense, the results from the static/polling scheduling problem indicate that FEDD is a good scheduling policy for the dy-

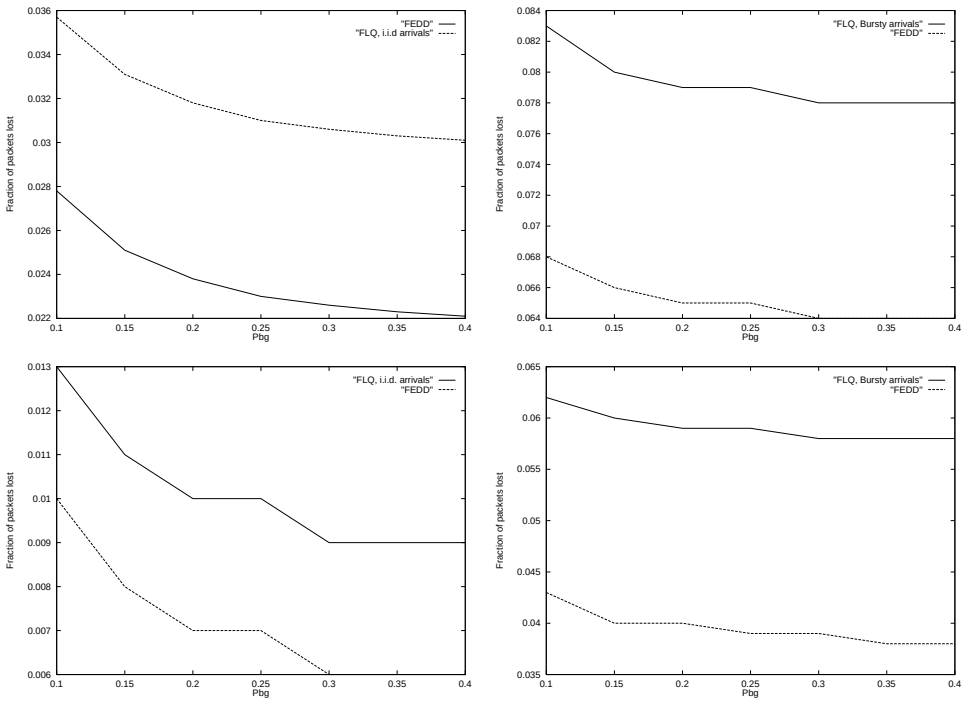

Figure 7. Simulation results: In these plots, Performance of FEDD versus the longest queue first policy. $P_{gb} = 0.001$.

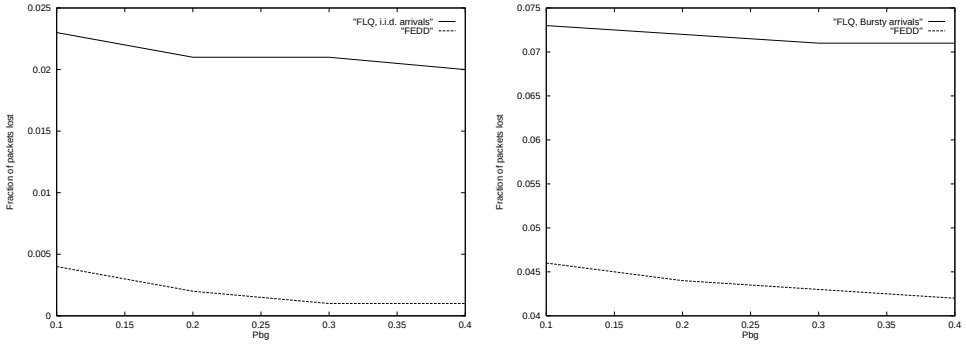

Figure 8. Simulation results: In these plots, Performance of FEDD versus the longest queue first policy for a 10 queue system. $P_{gb} = 0.001$. In the first plot, we study i.i.d. arrivals to the two queues whereas in the second plot, bursty arrivals are studied.

namic case too. In Figure 7, we compare the performance of FEDD against the feasible longest queue first policy in the dynamic case. The feasible longest queue

(FLQ) policy schedules the HOL packet from the longest queue whose channel is in the "Good" state. In Figures 7, we consider a simulation setup consisting of a system with 2 queues with ON-OFF Markov arrivals with the channel processes being bursty as in the static case. The simulations were carried out over 500,000 slots. In each of the plots, we compare the performance of FLQ and FEDD for varying P_{bg} with fixed $P_{gb} = 0.001$. In the top two plots, we study the case with short deadlines (i.e., the packet expires quickly) while the other two plots compare compare the performance of the two policies when the deadlines are moderately long. The two plots on the left deal with the case of i.i.d. arrivals with the average load to the system maintained at about 90% while the two plots on the right deal with the case of bursty arrivals with mean burst size of 10. In each of the four cases, we note the FEDD outperforms FLQ by margins of about 30%. This fits in well with the results have seen in the static scheduling case.

In Figure 8, we study a 10 queue system with bursty channels. As before, we study i.i.d. and bursty arrivals. We can see in the i.i.d. case that FEDD results in an order of magnitude improvement over FLQ. Hence, for all the cases considered, FEDD performs better that FLQ.

5.3. One-step Prediction

So far, we have assumed that the scheduler has perfect knowledge of the channel. However, in practice, the best that it can do is to estimate the current state of the channel given the previous history. In Table 1, we compare the performance by simulation of the FEDD policy with and without perfect channel knowledge. In one case, we assume that the scheduler has perfect knowledge of the channel, whereas in the other case, we assume that the scheduler knows only the previous state of the channel and *estimates* the current state of the channel using the one-step prediction suggested in [15]. If a packet transmission is unsuccessful, then the packet has to be retransmitted in the future. We perform the simulations for different values of P_{01} and P_{10}, the transition probabilities of the states of the sources (assumed to be Markovian ON-OFF). We can see from the table that there is very small difference in the expected number of packets lost. Figures 9 compare the performances of FLQ and FEDD with channel estimation for i.i.d. and bursty arrival processes. We can see that FEDD outperforms FLQ by margins similar to the ones observed with perfect channel knowledge. This leads us to believe that for channels with high correlation, it does not make too big

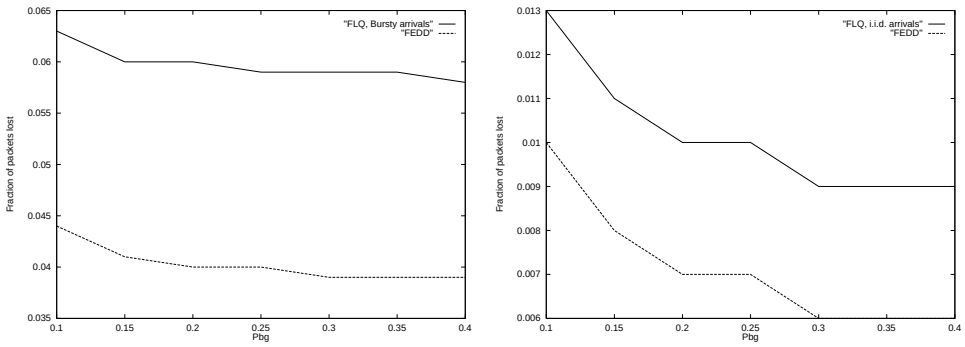

Figure 9. Simulation results: In this plot, Performance of FEDD versus the longest queue first policy with *channel state estimation*. $P_{gb} = 0.001$ and the sources are bursty (left) or i.i.d. (right).

a difference is estimation is used instead of perfect knowledge of the channel. This result is what we expect intuitively since channels with high correlations change slowly. Hence, most of the time the estimated value of the channel will actually be the correct state of the channel, which is consistent with the observations in [15].

5.4. WRR and FEDD

It is well-known that a deadline-based policy can be unfair without some form of traffic policing. Here, we assume that weighted round-robin (WRR) is used as an approximation to weighted fair-queuing to provide some degree of isolation between queues. Specifically, we consider two queues and a scheduling round consisting of 10 packets. During each round,in the first eight slots, a packet from each queue is scheduled alternately. The remaining two slots are given to

P_{01}	P_{10}	Perfect Knowledge	Estimation
0.2	0.8	0.001865	0.002092
0.025	0.1	0.113197	0.113668
0.0176	0.1	0.06445	0.064907

Table 1

Simulation Results: The simulations are run over 500000 slots. The source and channels are assumed to be of Markovian ON-OFF type. The simulations are performed for a 4-queue system, with each channel having $P_{gb} = 0.001$ and $P_{bg} = 0.1$. We compare the expected number of packets lost with perfect channel knowledge vs. channel estimation.

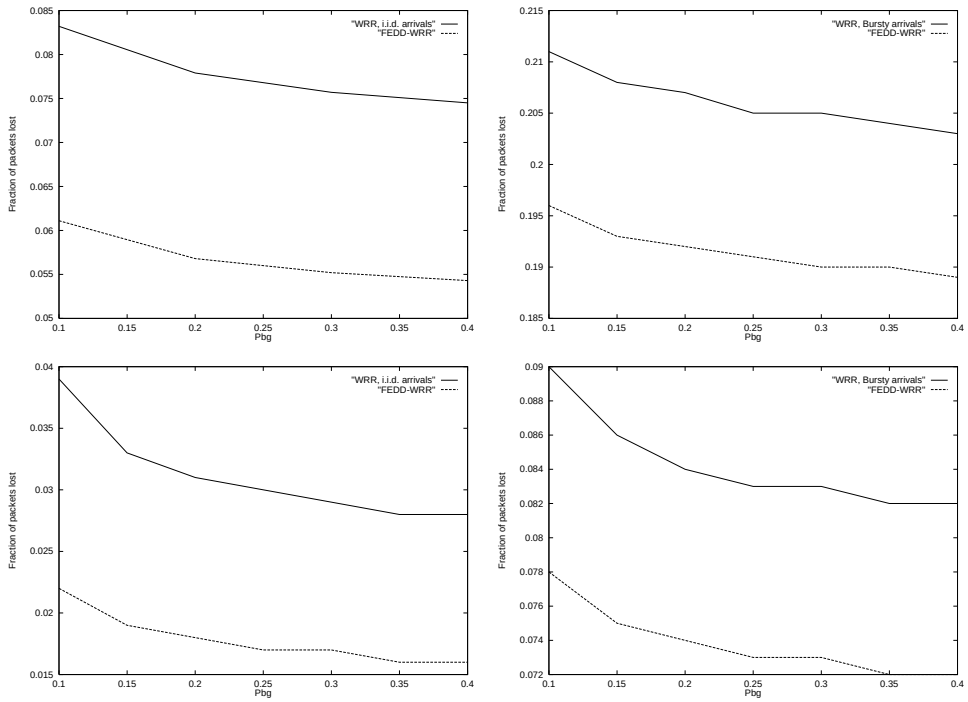

Figure 10. Comparison of WRR with and without FEDD compensation for i.i.d. and Bursty arrivals processes.

the packets with the earliest deadlines from either queue. Thus, during each round, each source is guaranteed at least two-fifths of the total bandwidth if its channel is error-free, thus providing some degree of isolation among the queues. The remain 1/5 of the bandwidth is allocated based on FEDD. This is similar, in spirit, to the policies proposed in [6] for wireline networks. In addition, this can also be thought as a scheme for compensating channels which perceive long bursts as in [15], [16] and [20]. Instead of using credits and debits, we use deadlines to compensate lagging flows. This, we feel, is a more natural method for choosing the magnitude of the compensation. The other methods do not prescribe a way for choosing the amount of compensation. In our scheme, if a flow perceives a long burst of error, then its deadlines would become smaller and it would use the two slots allocated for FEDD when its channel return to the Good state. We do not claim that the results here provide a complete solution to the general wireless scheduling problem with fairness and deadline considerations. However, the results here can prove to be a useful starting point for such a work. In

Figure 10, we compare WRR with and without the FEDD compensation scheme (FEDD-WRR) described above. As before, the top two figures compare the performance of WRR and FEDD-WRR for short deadlines and the lower two figures compare their performance for longer deadlines. The two plots on the left are for i.i.d. arrivals to the queues and the other two are for bursty arrivals (mean burst size of 10). Using FEDD to compensate for prolonged error bursts results in at least a 30% reduction in packet loss due to deadline expiry.

6. Conclusion

In this paper, we have studied the optimal scheduling problem for real time traffic with deadlines. We defined a modified version of the EDD policy, namely, the Feasible EDD policy, which schedules based on EDD over channels that are perceived to be in Good state. While this policy is not always optimal, we have shown that is optimal for a class of deterministic arrival processes. We then studied a static/polling scheduling problem, i.e., a problem with an initial number of packets in each queue and no further arrival. This led us to obtain insight into the general dynamic scheduling problem. For a large range of channel parameters and initial queue lengths and deadlines, the FEDD policy was the indeed optimal policy.

We then simulated a dynamic queueing system and saw that FEDD performed better than the longest-queue-first policy. In the analysis and first set of simulations, we assumed that the scheduler was aware of the current channel state. We then removed this assumption and found by simulations that this assumption is reasonable for channels with highly bursty error behavior. Finally, we showed how this policy can be combined with other policies to enhance its fairness properties.

Appendix

Appendix A: Proof of Proposition 4.1

The proof consists of a sequence of four arguments illustrated in Figure 11. We consider the following five scenarios:

- System 1: The deadline of the HOL packet of queue 1 is d, the deadline of the

HOL packet of queue 2 is $d-1$, the deadlines are separated by exactly t_a time units and the number of packets in queue 1 is L.

- System 2: The number of packets in queue 1 is $m \leq L$, everything else remaining the same as System 1.
- System 3: We relax the inter-arrival times between the packets in queue 1 to be $\geq t_a$, the number of packets in each queue is the same as in System 2.
- System 4: The deadlines of the HOL packets in queue 2 and queue 1 are $d_2(1)$ and $d_2(1)+1$, respectively, where $d_2(1) \leq d-1$. The inter-arrival times and number of packets in each queue in the same as System 3.
- System 5: Finally, queue 1's HOL packet's deadline is $d_1(1) = d_2(1)+k, k \geq 1$, which gives us the situation in Figure 2.

To prove the theorem, we have to show that, if FEDD is optimal for System 1, then it is also optimal for System 5. Instead, we can do this in a sequence of steps as follows: for each i, $i = 1, 2, 3, 4$ prove that, if FEDD is optimal for System i, then it is also optimal for System $i+1$.

In what follows, we show that FEDD is optimal in System 2 if FEDD is optimal in System 1. The rest of the steps follow in a similar manner. We first look at System 1. Let Ω be the set of all *channel sample paths* over the interval of time $[1, d+(L-1)t_a]$.

Define $\mathcal{F}$ to be the policy which schedules the HOL packet from queue 1 initially and for all future times, follows the FEDD policy. Further, for *any* policy $\mathcal{P}$, define

$\mathcal{E}_{\mathcal{P}} =$ *expected* number of packets lost when the scheduling policy is $\mathcal{P}$

$\mathcal{N}_{\mathcal{P}}^{\omega} =$ number of packets lost when the scheduling policy is $\mathcal{P}$ on the sample path ω

By assumption, FEDD is an optimal scheduling policy in System 1, i.e., $\mathcal{E}_{\text{FEDD}} - \mathcal{E}_{\mathcal{F}} \leq 0$. Expanding this, we have

$$\mathcal{E}_{\text{FEDD}} - \mathcal{E}_{\mathcal{F}} = \sum_{\omega \in \Omega_{\text{FEDD}}} (\mathcal{N}_{\text{FEDD}}^{\omega} - \mathcal{N}_{\mathcal{F}}^{\omega}) Pr(\omega) + \sum_{\omega \in \Omega_{\text{FEDD}}^{c}} (\mathcal{N}_{\text{FEDD}}^{\omega} - \mathcal{N}_{\mathcal{F}}^{\omega}) Pr(\omega) \leq 0 \quad (10)$$

where $\Omega_{\text{FEDD}} \subseteq \Omega$ is the set of all sample paths ω over which the number of packets lost due to FEDD is no greater than that due to $\mathcal{F}$. We note the following properties:

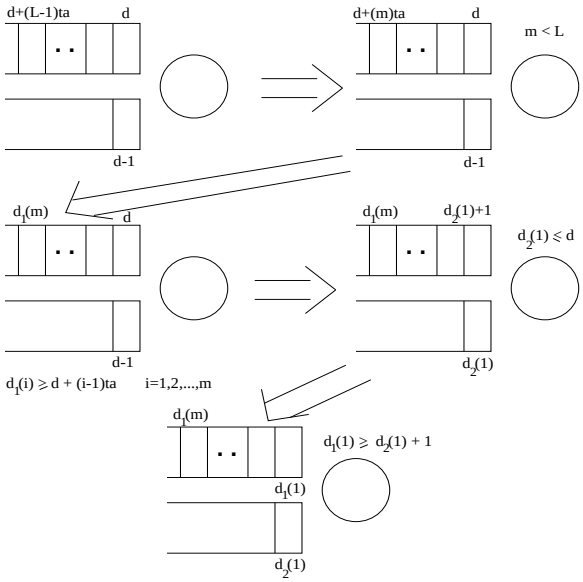

Figure 11. Proof of Proposition 4.1: "Worst case" performance of FEDD occurs in System 1 where queue 1 has L packets, $d_1 = d, d_2 = d-1$ and deadlines of packets in queue 1 are separated by exactly t_a.

- As there is only 1 packet in queue 2, over sample paths $\omega \in \Omega^c_{\text{FEDD}}$, we have $(\mathcal{N}^{\omega}_{\text{FEDD}} - \mathcal{N}^{\omega}_{\mathcal{F}}) = 1$. This follows because in the worst case, only one slot is lost to queue 1 which can result in at most 1 extra packet loss.
- Over those sample paths over $\omega \in \Omega_{\text{FEDD}}$, we have $(\mathcal{N}^{\omega}_{\text{FEDD}} - \mathcal{N}^{\omega}_{\mathcal{F}}) = 0$ or -1. This follows as $\mathcal{F}$ follows the FEDD policy for all future times.

Hence, we can rewrite the above equation as

$$\mathcal{E}_{\text{FEDD}} - \mathcal{E}_{\mathcal{F}} = \sum_{\omega \in \Omega_{\text{FEDD}}} (\mathcal{N}^{\omega}_{\text{FEDD}} - \mathcal{N}^{\omega}_{\mathcal{F}}) Pr(\omega) + \sum_{\omega \in \Omega^c_{\text{FEDD}}} Pr(\omega) \leq 0 \quad (0.2)$$

Next, consider *System 2*. We look at the expected number of packets lost here. This is given by

$$\mathcal{E}_{\text{FEDD}} - \mathcal{E}_{\mathcal{F}} = \sum_{\widehat{\omega}} (\mathcal{N}^{\widehat{\omega}}_{\text{FEDD}} - \mathcal{N}^{\widehat{\omega}}_{\mathcal{F}}) Pr(\widehat{\omega}) \quad (0.3)$$

$$= \sum_{\omega} (\mathcal{N}^{\omega}_{\text{FEDD}} - \mathcal{N}^{\omega}_{\mathcal{F}}) Pr(\omega) \quad (0.4)$$

$$= \sum_{\omega \in \widehat{\Omega}_{\text{FEDD}}} (\mathcal{N}^{\omega}_{\text{FEDD}} - \mathcal{N}^{\omega}_{\mathcal{F}}) Pr(\omega) + \sum_{\omega \in \widehat{\Omega}^c_{\text{FEDD}}} Pr(\omega) \quad (0.5)$$

where ω are the set of channel sample paths over $[1, d + (L-1)t_a]$ and $\widehat{\omega}$ are the set of channel sample paths over $[1, d + (m)t_a]$. This equality follows from the fact that all packets expire by $d + (m)t_a$ and appending extra slots make no difference to the expected number of lost packets.

Let compare the two summations in Equations 0.2 and 0.5. Consider any sample path $\omega \in \widehat{\Omega}^c_{\text{FEDD}}$. Over the same sample path ω in System 1, we can get the same gain of 1 packet simply by following the same scheduling sequence (which corresponds to the same policy as the channel sample paths and the initial conditions are the same) as in System 2. Hence, the second summation in Equation 0.2 is no lesser than that in Equation 0.5.

Let us now consider the first summation. $(\mathcal{N}^{\omega}_{\text{FEDD}} - \mathcal{N}^{\omega}_{\mathcal{F}}) = -1$ can occur only if the channel of queue 2 turns "Bad" at $t = 2$ and remains "Bad" till $d-1$. This follows from the fact the policy $\mathcal{F}$ will be the same as the FEDD policy from the next time slot. Hence, $w \in \Omega \Rightarrow w \in \widehat{\Omega}$. Hence, the first summation in System 2 is *at least as small as* that in System 1.

$$\mathcal{E}_{\text{FEDD}} - \mathcal{E}_{\mathcal{F}} \leq 0 \text{ for } l = L \quad \Rightarrow \quad \mathcal{E}_{\text{FEDD}} - \mathcal{E}_{\mathcal{F}} \leq 0 \ \ \forall m \leq L \tag{0.6}$$

So far, we have compared FEDD with $\mathcal{F}$. We next generalize the above result to hold over *all* causal policies $\tilde{\mathcal{F}}$. First consider the case when $m = 1$. From Lemma 3.1, we have

$$\mathcal{E}_{\text{FEDD}} - \mathcal{E}_{\tilde{\mathcal{F}}} \leq 0 \tag{0.7}$$

where $\tilde{\mathcal{F}}$ is any other causal scheduling policy. Hence, for $m = 1$, FEDD is the optimal scheduling policy. When $m = 2$, Equations 0.6 and 0.7 together imply that FEDD is better than any other policy. We argue inductively in a similar manner $\forall m \leq L$. ∎

With the above notation, we note that Corollary 4.1 states that System 1 is optimal implies System 4 is optimal. This is an immediate consequence of the above proof. Corollary 4.2 basically restates the induction step in the proof above for System 1.

Appendix B: Recursive Equations for Loss Computation

Let us define $\mathcal{E}_l(t, k, l, S_1, S_2)$ as the expected number of packets lost from time $t, 0 \leq t \leq D_{max}$ with k packets in queue 1, l packets in queue 2 and the

state of the channels of the two queues being $S_1, S_2 \in \{\mathrm{G}, \mathrm{B}\}$, where G and B correspond to the channel state at time t being "Good" or "Bad". Further, we define d_1 and d_2 as the times to expiry for the HOL packets of queues 1 and 2 respectively (given that k and l are ≥ 1). Then, at time t,

$$d_1 = D_{max} - t - (k-1)t_a \tag{0.8}$$

$$d_2 = (D_{max} - t - (K-1)t_a - 1) \tag{0.9}$$

Then, for $1 \leq t \leq D_{max}$, if the channel state at t is (G, G),

$$\mathcal{E}_l(t,k,l,G,G) = \begin{cases} P_{gg}^2 \mathcal{E}_l(t+1,k,0,G,G) + P_{gg}P_{gb}\mathcal{E}_l(t+1,k,0,G,B) \\ +P_{gb}P_{gg}\mathcal{E}_l(t+1,k,0,B,G) + P_{gb}^2\mathcal{E}_l(t+1,k,0,B,B) & \text{if } l=1 \\ & \\ P_{gg}^2 \mathcal{E}_l(t+1,k-1,0,G,G) + P_{gg}P_{gb}\mathcal{E}_l(t+1,k-1,0,G,B) \\ +P_{gb}P_{gg}\mathcal{E}_l(t+1,k-1,0,B,G) + P_{gb}^2\mathcal{E}_l(t+1,k-1,0,B,B) & \text{if } l=0 \end{cases} \tag{0.10}$$

If the state at t is (G, B),

$$\mathcal{E}_l(t,k,l,G,B) = \begin{cases} \begin{cases} 1 + P_{gg}P_{bg}\mathcal{E}_l(t+1,k-1,0,G,G) + P_{gg}P_{bb}\mathcal{E}_l(t+1,k-1,0,G,B) \\ +P_{gb}P_{bg}\mathcal{E}_l(t+1,k-1,0,B,G) + P_{gb}P_{bb}\mathcal{E}_l(t+1,k-1,0,B,B) & \text{if } d_2 = 1 \end{cases} & \text{if } l=1 \\ \begin{cases} P_{gg}P_{bg}\mathcal{E}_l(t+1,k-1,1,G,G) + P_{gg}P_{bb}\mathcal{E}_l(t+1,k-1,1,G,B) \\ +P_{gb}P_{bg}\mathcal{E}_l(t+1,k-1,1,B,G) + P_{gb}P_{bb}\mathcal{E}_l(t+1,k-1,1,B,B) & \text{if } d_2 > 1 \end{cases} & \\ & \\ P_{gg}P_{bg}\mathcal{E}_l(t+1,k-1,0,G,G) + P_{gg}P_{bb}\mathcal{E}_l(t+1,k-1,0,G,B) \\ +P_{gb}P_{bg}\mathcal{E}_l(t+1,k-1,0,B,G) + P_{gb}P_{bb}\mathcal{E}_l(t+1,k-1,0,B,B) & \text{if } l=0 \end{cases} \tag{0.11}$$

We note that the various cases above arise due to the fact that if the channel of queue 2 is "bad" at t and the HOL packet expires at the end of the current slot, then that packet is lost due to deadline expiry.

If the state at t is (B, G),

$$\mathcal{E}_l(t,k,l,B,G) =$$

$$\begin{cases} P_{bg}P_{bg}\mathcal{E}_l(t+1,k,0,G,G) + P_{bg}P_{gb}\mathcal{E}_l(t+1,k,0,G,B) \\ +P_{bb}P_{gg}\mathcal{E}_l(t+1,k,0,B,G) + P_{bb}P_{gb}\mathcal{E}_l(t+1,k,0,B,B) & \text{if } l=1 \\ \\ \begin{cases} 1+P_{bg}P_{gg}\mathcal{E}_l(t+1,k-1,0,G,G) + P_{bg}P_{gb}\mathcal{E}_l(t+1,k-1,0,G,B) \\ +P_{bb}P_{gg}\mathcal{E}_l(t+1,k-1,0,B,G) + P_{bb}P_{gb}\mathcal{E}_l(t+1,k-1,0,B,B) & \text{if } d_1=1 \\ & \text{if } l=0 \\ P_{bg}P_{gg}\mathcal{E}_l(t+1,k,0,G,G) + P_{bg}P_{gb}\mathcal{E}_l(t+1,k,0,G,B) \\ +P_{bb}P_{gg}\mathcal{E}_l(t+1,k,0,B,G) + P_{bb}P_{gb}\mathcal{E}_l(t+1,k,0,B,B) & \text{if } d_1>1 \end{cases} \end{cases} \tag{0.12}$$

As in the earlier case, the various cases arise depending on whether a packet is lost due to deadline expiry in queue 1. Note that in the case where $l=1$, we need not consider the case where both the HOL packets can expire as the fact that $l=1$ means that the expiry times of the packets in queue 1 are *at least* 2 units of time away in the future.

Finally, if the state at t is (B,B),

$$\mathcal{E}_l(t,k,l,B,B) =$$
$$\begin{cases} \begin{cases} 1+P_{bg}^2\mathcal{E}_l(t+1,k,0,G,G) + P_{bg}P_{bb}\mathcal{E}_l(t+1,k,0,G,B) \\ +P_{bb}P_{bg}\mathcal{E}_l(t+1,k,0,B,G) + P_{bb}^2\mathcal{E}_l(t+1,k,0,B,B) & \text{if } d_2=1 \\ & \text{if } l=1 \\ P_{bg}^2\mathcal{E}_l(t+1,k,1,G,G) + P_{bg}P_{bb}\mathcal{E}_l(t+1,k,1,G,B) \\ +P_{bb}P_{bg}\mathcal{E}_l(t+1,k,1,B,G) + P_{bb}^2\mathcal{E}_l(t+1,k,1,B,B) & \text{if } d_2>1 \end{cases} \\ \\ \begin{cases} 1+P_{bg}^2\mathcal{E}_l(t+1,k-1,0,G,G) + P_{bg}P_{bb}\mathcal{E}_l(t+1,k-1,0,G,B) \\ +P_{bb}P_{bg}\mathcal{E}_l(t+1,k-1,0,B,G) + P_{bb}^2\mathcal{E}_l(t+1,k-1,0,B,B) & \text{if } d_1=1 \\ & \text{if } l=0 \\ P_{bg}^2\mathcal{E}_l(t+1,k,0,G,G) + P_{bg}P_{bb}\mathcal{E}_l(t+1,k,0,G,B) \\ +P_{bb}P_{bg}\mathcal{E}_l(t+1,k,0,B,G) + P_{bb}^2\mathcal{E}_l(t+1,k,0,B,B) & \text{if } d_1>1 \end{cases} \end{cases} \tag{0.13}$$

The recursion terminates at $k=0$. If initially, there are K packets in queue 1, then, the expected number of packets lost is given by

$$\begin{aligned} E_{\mathcal{F}}(\text{lost}) = {} & P_{gg}^2\mathcal{E}_l(1,K,1,G,G) + P_{gg}P_{gb}\mathcal{E}_l(1,K,1,G,B) \\ & +P_{gb}P_{gg}\mathcal{E}_l(1,K,1,B,G) + P_{gb}^2\mathcal{E}_l(1,K,1,B,B) \end{aligned} \tag{0.14}$$

References

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

[2] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, "MACAW: A Media Access Protocol for Wireless LANs." *ACM Sigcomm '94*, 1994.

[3] P. Bhattacharya and A. Ephremides, "Optimal Scheduling with Strict Deadlines", *IEEE Tran. Auto. Control*, Vol. 34, No. 7, July 1989.

[4] R. Cruz, "A Calculus for Network Delay, Part I: Network Elements in Isolation", *IEEE Tran. Info. Theory*, Vol. 37, No. 1, January 1991.

[5] A. Demers, S. Keshav and S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm", *Internetworking: Research and Experience*, pp. 3-26, Vol. 1, 1990.

[6] N.G. Duffield, T.V. Lakshman and D. Stiliadis, "On Adaptive Bandwidth Sharing with Rate Guarantees", *Proc. IEEE Infocom'98*, April 1998.

[7] N. Figueira and J. Pasquale, "A Schedulability Condition for Deadline-Based Service Disciplines", *IEEE/ACM Tran. on Networking*, Vol. 5, No. 2, April 1997.

[8] C. Fragouli, V. Sivaraman and M. Srivastava, "Controlled multimedia wireless link sharing via enhanced class-based queueing with channel-state-dependent packet scheduling" , *Proc. IEEE Infocom'98*, April 1998.

[9] L. Georgiadis, R. Guerin and A. Parekh, "Optimal Multiplexing on a Single Link: Delay and Buffer Requirements", *IEEE Tran. Info. Theory*, Vol. 43, No. 5, September 1997.

[10] S. Golestani, "A Self-Clocked Fair Queueing Scheme for Broadband Applications", *Proc. IEEE Infocom'94*, June 1994.

[11] P. Goyal, H. Vin and H. Cheng, "Start-time Fair Queueing: A Scheduling Algorithm for Integrated Services Packet Switching Networks", *Proc. ACM Sigcomm'96*, August 1996.

[12] B. Hajek and P. Seri, "On Causal Scheduling of Multiclass Traffic with Deadlines" *Proc. IEEE ISIT'98*, August 1998.

[13] W. C. Jakes. *Microwave Mobile Communications*. IEEE Press, 1993.

[14] G. Koole, Z. Liu and R. Righter, "Optimal Transmission Policies for Noisy Channels", preprint.

[15] S. Lu, V. Bharghavan and R. Srikant, "Fair Scheduling in Wireless Packet Networks", *Proc. ACM Sigcomm'97*, September 1997.

[16] T.S.E. Ng , I. Stoica and H. Zhang, "Packet fair queueing algorithms for wireless networks with location-dependent errors", *Proc. IEEE Infocom 98*, April 1998.

[17] S. Panwar, D. Towsley and J. Wolf, "Optimal scheduling policies for a class of queue with customer deadlines to the beginning of services", *Journal of ACM*, Vol. 35, No. 4, pp. 832-844, 1988.

[18] A. Parekh and R. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case", *IEEE/ACM Tran. on Networking*, Vol. 1, No. 3, June 1993.

[19] A. Parekh and R. Gallager, "A Generalized Processor Sharing Approach to Flow Control in

Integrated Services Networks: The Multiple Node Case", *IEEE/ACM Tran. on Networking*, Vol. 2, No. 2, April 1994.

[20] P. Ramanathan and P. Agrawal, "Adapting Packet Fair Queueing Algorithms to Wireless Networks", *Proc. ACM Mobicom'98*, 1998.

[21] L. Tassiulas and A. Ephremides, "Dynamic Server Allocation to Parallel Queue with Randomly Varying Connectivity", *IEEE Tran. Info. Theory*, Vol. 30, No. 2, March 1993.

[22] L. Zhang, "Virtual Clock: A New Traffic Control Algorithm for Packet Switching Networks", *ACM Tran. Comput. Syst.*, Vol. 9, May 1991.