SCHEDULING MULTICLASS PACKET STREAMS TO MINIMIZE WEIGHTED LOSS*

Robert L. Givan Edwin K. P. Chong
Hyeong Soo Chang
School of Electrical and Computer Engineering
Purdue University, West Lafayette, IN 47907-1285
E-mail: {givan,echong,hyeong}@ecn.purdue.edu

January 24, 2001

Abstract

We consider the problem of scheduling an arriving sequence of packets at a single server. Associated with each packet is a deadline by which the packet must be scheduled. Each packet belongs to one of a predetermined set of classes, and each class has an associated weight value. The goal is to minimize the total weighted value of the packets that miss their deadlines. We first prove that there is no policy that minimizes this weighted loss for all finite arrival sequences of packets. We then present a class of greedy scheduling policies, called the *current-minloss throughput-optimal* (CMTO) policies. We characterize all CMTO policies, and provide examples of easily implementable CMTO policies. We compare CMTO policies with a multiclass extension of the earliest-deadline-first (EDF) policy, called EDF+, establishing that a subclass of CMTO policies achieves no more weighted loss than EDF+ for any traffic sequence, and at the same time achieves a substantial weighted-loss advantage over EDF+ for some traffic sequences—this advantage is shown to be arbitrarily close to the maximum possible achievable advantage. We also provide empirical results to quantify the weighted-loss advantage of CMTO policies over EDF+ and the static-priority (SP) policy, showing an advantage exceeding an order of magnitude when serving heavy-tailed aggregations of MPEG traces.

Keywords: multiclass, deadlines, scheduling, weighted loss, throughput optimality, EDF, SP

*This research is supported by DARPA/ITO under contracts F19628-98-C-0051 and F30602-00-2-0552. The equipment for this work was provided in part by a grant from Intel Corporation. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U. S. Government.

1 Introduction

We consider the problem of scheduling multiclass traffic with deadlines under a weighted-loss criterion. In this problem, a single server receives packet traffic in discrete time. Each packet has an associated deadline, and must be served by the deadline, or it is considered lost. Each packet belongs to one of a predetermined finite set of classes, and each class has a real-number weight corresponding to the cost incurred for each packet lost in that class. Every packet takes one unit of time to serve, and any finite number of packets can arrive per unit time. At each time, the server makes a decision on which packet to serve based on the packets pending in the buffer (i.e., in an "on-line" fashion). The problem of scheduling packets with deadlines has received considerable interest in the literature; e.g., [11, 12, 13, 14, 15, 17, 16, 18]. However, the treatment of on-line scheduling problems with multiple classes of traffic remains relatively incomplete. Our main contribution in this paper is the description of a new family of multiclass scheduling policies, a characterization of its properties, and a rigorous analytical comparison of its performance relative to a multiclass extension of the *earliest-deadline-first* (EDF) policy. We also provide substantial empirical results demonstrating significant superiority for the new scheduling policies over both EDF and the static priority (SP) policy—these results suggest that realistic multiclass traffic with deadlines requires a policy that explicitly considers the interplay between class and deadline (rather than focusing on one dimension and using the other to break ties).

The multiclass aspect of our problem is motivated by "differentiated" packet delivery in high-speed computer networks. There has been related work by the Internet Engineering Task Force (IETF) *intserv* and *diffserv* communities, aiming at extending the best-effort service offered by the current Internet in order to provide service differentiation beyond the traditional single-class packet delivery service. For example, applications with layered coding of data, such as Moving Picture Expert Group (MPEG) video streams, layered Discrete Cosine Transform (DCT), or wavelet coders require packets from "important" layers to be delivered preferably, but can tolerate some losses of the packets from "unimportant" layers. In an MPEG stream, the I frames are considered more important than the P or B frames [1].

Several new approaches have been explored in the networking community to address the problem of service differentiation. One approach is to use a simple FIFO queue for the aggregated traffic and provide service differentiation by applying different dropping preferences to tagged or untagged packets [7]. Another approach is to distinguish between packets by assigning to each packet an associated class and then apply a multiclass scheduling algorithm, e.g., *static-priority* (SP), which always schedules a highest-class packet in the queue. A third approach is to use "pricing" to differentiate the services [4]. Selection of the best approach is still on-going [9]. Generally, each of these approaches can be handled (to varying degrees of precision) by introducing *class weights*. For example, the weight of a class can be interpreted as the per-packet price [4], per-class dropping

utility, or a relative packet importance measure.

We consider scheduling policies that decide which packet to transmit based solely on the packets currently pending in the buffer. The earliest-deadline-first (EDF) policy, which at each time selects a packet with the earliest deadline among the packets in the buffer, is a well-known example of such a policy. EDF is a *throughput optimal* (TO) policy in the sense that it serves as many packets over any time interval as any other policy for *any traffic*. This TO characterization of EDF provides a deterministic, traffic-independent statement of the throughput of EDF. In [15], Ling and Shroff describe a policy that also achieves optimal throughput in the above sense, but drops some packets strictly before their deadlines. The recent work of Hajek and Seri [11, 12] provides a rigorous study of throughput optimal policies using a general deterministic, traffic-independent optimality framework. Their work also includes a treatment of the multiclass setting under an optimality criterion called *lex-optimality* [12].

In this paper, we build on the deterministic, traffic-independent framework of Hajek and Seri by considering the multiclass criterion of *weighted loss*. Specifically, we are interested in minimizing the total weight of lost packets. It turns out that there is no traffic-independent optimal policy; i.e., there is no policy that has minimum weighted loss for all traffic sequences. The same is true even if we restrict our attention only to the class of TO policies—so there is in some sense no "best" TO policy with respect to weighted loss. Our approach is to consider a subclass of TO policies, called CMTO policies, that take into account the weights of packets in the buffer according to a "greedy" scheme called *current-minloss* scheduling. Specifically, a CMTO policy is a throughput optimal policy that selects a packet to serve at each time consistent with the goal of minimizing the weighted loss assuming that there are no further arrivals. Minimizing the weighted loss of packets currently in the queue is in a sense the best we can hope for, without any assumptions on the future traffic.

We characterize all CMTO policies by providing a necessary and sufficient condition on the choice of packet to serve that such policies must make at each time step. It turns out that this characterization suggests a natural two-step implementation of CMTO policies. In Step 1, the policy limits its attention to a subset of packets in the buffer (called an "eligible set"), and in Step 2 the policy selects a packet from the eligible set in a way that ensures throughput optimality. We present easily implementable algorithms for both steps, yielding simple example instances of CMTO policies.

We compare the CMTO policy with a simple multiclass extension of EDF, called EDF+ that uses packet class only to break deadline ties, revealing the clear advantage of CMTO policies over EDF+. Specifically, we show that a subclass of the CMTO policies achieves no more weighted loss than EDF+ for *any* traffic. This result makes a deterministic, traffic-independent statement about the weighted loss of CMTO policies in relation to EDF+ for any possible traffic arrival sequence. We also show that there exist traffic sequences such that any CMTO policy achieves a weighted-loss

advantage over EDF+ that is arbitrarily close to the maximum possible achievable advantage.

We also describe two extensions of CMTO policies. The first extension involves dropping some packets from the buffer strictly before their deadlines. Such "scheduling-dropping" policies are appealing because of their smaller buffer requirements and their ability to provide more timely feedback on which packets will not be scheduled. We show that a natural subclass of CMTO policies can be extended to scheduling-dropping policies without changing the schedule of packets served, while at the same time minimizing the buffer usage among all throughput optimal scheduling-dropping policies. The second extension involves a credit-based mechanism to incorporate fair link-sharing into CMTO policies. The result is a family of class- and deadline-sensitive scheduling policies that explicitly takes into account link-share allocations for individual calls. Our preliminary empirical results, presented below, show that these policies retain the substantial advantages of CMTO policies with respect to weighted loss while giving fair link-sharing behavior similar to that of previously proposed link-sharing schemes (e.g. [10]).

The remainder of this paper is organized as follows. In Section 2, we introduce our notation and terminology, give a precise problem definition, and show the non-existence of an optimal scheduling policy. In Section 3, we describe the class of CMTO scheduling policies. We present a necessary and sufficient condition characterizing all CMTO policies. We also provide examples of easily implementable CMTO policies, with correctness proofs and algorithmic complexity analyses. In Section 4, we analytically compare CMTO policies and a multiclass extension of the EDF policy (called EDF+), illustrating the clear advantage of CMTO over EDF+. We then describe some extensions of CMTO policies in Section 5: CMTO policies that drop packets before their deadlines, and policies based on CMTO that also incorporate link-share fairness. In Section 6, we illustrate the performance of CMTO policies via simulations using practically reasonable traffic sequences, comparing the performance of CMTO with EDF+, SP, and an SP-based fair-queueing policy. We draw conclusions in Section 7 and also provide there some discussion of possible future research directions. Throughout the paper, we relegate technically involved proofs to the appendix.

2 Multiclass Scheduling Problem

In this section, we present the notation and terminology that we will use throughout the paper, and demonstrate the non-existence of a policy that minimizes weighted loss under all traffic. Our treatment follows that of Hajek and Seri [11] (although some of our notation and terminology differs from [11]).

2.1 Framework, terminology, and notation

We assume that time t is slotted, i.e., $t \in \{1, 2, \ldots\}$, and that each packet takes exactly one time slot to be served. Each packet p belongs to a particular class, denoted by $C(p) \in \{1, \ldots, m\}$, and

has a deadline $d(p) \in \{1, 2, \ldots\}$. A packet that is not served by its deadline is said to be lost. More precisely, if the arrival time for a packet p is t, then p needs to be served at some time between t and $d(p)$, inclusive, or p is lost. If p is lost, a cost $w_{C(p)} > 0$, depending only on the class of p, is incurred. We assume without loss of generality that if $C(p) < C(p')$, then $w_{C(p)} > w_{C(p')}$. We naturally consider that high-weight classes are more important than low-weight ones.

An *arrival sequence* (also called a *traffic sequence*, or a *traffic* for simplicity) A is a sequence $A = \{A_t : t \geq 1\}$ such that A_t is the set of packets arriving at time t. The sets A_t are assumed to be disjoint, and $d(p) \geq t$ for all $p \in A_t$. The *laxity* of a packet p at time t is defined as $l_t(p) = d(p) - t + 1$, i.e., the number of time slots left before the packet's deadline expires and p becomes lost. An arrival sequence is said to be *finite* if there is a finite T such that $A_t = \emptyset$ for all $t \geq T$. For convenience, we also use the notation A to denote the set of packets $\cup_{t \geq 1} A_t$.

Given an arrival sequence A, a *schedule* is a one-to-one partial mapping $t \mapsto p_t$ where $t \in \{1, 2, \ldots\}$ and, if p_t is defined, $p_t \in A_1 \cup \cdots \cup A_t$ with $t \leq d(p_t)$. If p_t is defined, we say that p_t is *scheduled* (alternatively, *transmitted* or *served*) at time t.

A *scheduling policy* π is a sequence of maps $\{\pi_t : t \geq 1\}$ where π_t is a function that maps any nonempty set of packets P to a single element of P, where we require that the maps $\{\pi_t\}$ are *stationary*; i.e., the function π_1 must determine π_t for all t as follows: for any set of packets P, $\pi_t(P) = \pi_1(P')$, where the packets of P' are exactly those of P but with their deadlines decreased by $t-1$ time steps (packets with nonpositive resulting deadlines are removed). A scheduling policy induces a schedule as follows. Given an arrival sequence A and a policy π, define the sequence $\{P_t\}$ by $P_0 = \emptyset$ and, for all $t \geq 0$,

$$P_{t+1} = A_{t+1} \cup P_t - \{p \in P_t : d(p) = t \text{ or } p = \pi_t(P_t)\}\,.$$

Note that the sequence $\{P_t\}$ depends on and is completely determined by the arrival sequence A and the policy π. At each time t, P_t represents the set of packets that have arrived at or before time t, have not been transmitted before time t, and have not yet missed their deadlines at time t. We refer to packets in P_t as *available* or *pending* in the system. We can also think of P_t as the state of the queue at time t, in the case where no packets are dropped ahead of their deadlines expiring. (We assume that the queue starts out empty: $P_0 = \emptyset$.) The policy π selects the packet $\pi_t(P_t)$ to transmit at time t. In other words, the scheduling policy π induces the schedule $t \mapsto \pi_t(P_t)$. We say that π *schedules* a packet p at time t if $\pi_t(P_t) = p$. Note that if P_t is empty, $\pi_t(P_t)$ is undefined, and no packet is served at time t.

A policy defined as above is often referred to as *work-conserving* or *non-idling* because $\pi_t(P_t)$ is defined whenever $P_t \neq \emptyset$, and as *causal* because its selection of the packet to transmit depends only on the currently pending packets in the system. Our definition of a policy is more restrictive than the one in [11], but suffices for our purposes (the definition in [11] allows for packet selections that depend arbitrarily on all past arrivals, not just the pending packets).

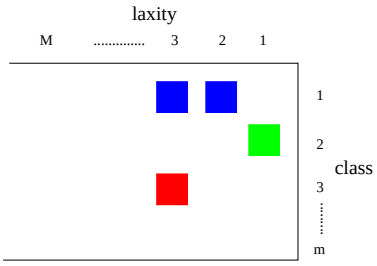

Figure 1: Graphical illustration of multiclass packets with deadlines

Throughout this paper, we graphically illustrate sets of packets using rectangular arrays, such as the one shown in Figure 1. The columns of the array represent laxities, ordered from right to left in increasing order, while the rows represent classes, ordered from top to bottom in increasing order (decreasing importance). A block in an entry of the array represents one or more packets (as labeled) with the associated laxity and class—if multiple packets have the same laxity and class, we indicate the number of packets above or below the appropriate block.

We need a few more definitions before we can describe our problem. The next definition is taken directly from [11]. A scheduling policy π is *throughput optimal* (TO) if, for any arrival sequence A and any $t \geq 1$, policy π schedules at least as many packets in slots $\{1, \ldots, t\}$ as any other policy does.

Denote the set of packets scheduled by π as S^π; i.e., $S^\pi = \{\pi_t(P_t) : P_t \neq \emptyset,\ t \geq 1\}$. Note that $A - S^\pi$ is the set of packets that are not served by π and will eventually miss their deadlines. The *weighted loss* incurred by a scheduling policy π with respect to a finite arrival sequence A is

$$L^\pi(A) = \sum_{p \in A - S^\pi} w_{C(p)}.$$

We say that a scheduling policy π *dominates* π' *over* A if $L^\pi(A) \leq L^{\pi'}(A)$. If π dominates π' over all finite arrival sequences, then we say that π dominates π'. Note that an equivalent definition of domination can be stated using the notion of *weighted throughput*, the total weight of packets served.

A scheduling policy π is said to be *optimal* if π dominates any other policy. If π is TO and dominates any other TO policy, we say that π is an *optimal-TO* (OTO) policy. Note that an OTO policy need not be optimal, because it is required to dominate only other TO policies.

2.2 Nonexistence of optimal policies

Ideally, our goal in multiclass scheduling should be to find an optimal policy as defined above. However, it is easy to show that there is no such policy.

Proposition 1 *No optimal scheduling policy exists if and only if there are at least two classes.*

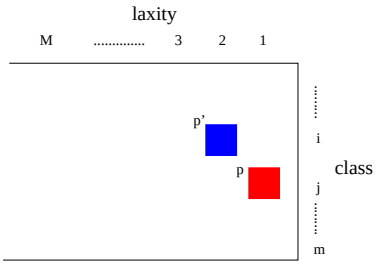

Figure 2: Graphical illustration for the proof of Proposition 1

Proof: If part: Let i and j be two classes with $w_i > w_j$. Suppose A_1 consists of a class j packet p with laxity $l_1(p) = 1$ and a class i packet p' with laxity $l_1(p') = 2$ (see Figure 2). If $A_t = \emptyset$ for all $t > 1$, we must serve p at time 1 to dominate all other policies over A. On the other hand, if A_2 consists of a class i packet with laxity 1, and $A_t = \emptyset$ for all $t > 2$, we need to serve p' at time 1 to dominate all other policies over A. Hence, no optimal policy exists.

Only if part: If all packets have the same weight, then any TO policy (e.g., the earliest-deadline-first (EDF) policy) is optimal. ∎

Even though there is in general no optimal policy, we might still hope for an OTO policy, since such a policy must only dominate all other TO policies, not necessarily all other policies. In the example (see Figure 2) used in the proof of Proposition 1, we described two options in choosing the packet to serve such that the preferable option depends on the future arrivals. However, only one of those options is consistent with throughput optimality, so this example does not immediately preclude the existence of an OTO policy.

To explore further the existence of OTO policies, we first review a throughput optimality condition by Hajek and Seri [11]—this condition is stated in Theorem 1 below, using the following notation. Given a nonempty set of packets P at time t (with deadlines no less than t) and integer laxity $l \geq 1$, let $N_t^l(P)$ be the number of packets in P having laxity no more than l, and let $\delta_t^l(P) = N_t^l(P) - l$. We say that P is *t-saturated* if $\delta_t^l(P) \geq 0$ for some $l \geq 1$. Define

$$h(P) = \begin{cases} \arg\max_l \delta_t^l(P) & \text{if } P \text{ is } t\text{-saturated} \\ \min\{l : N_t^l(P) > 0\} & \text{otherwise} \end{cases}$$

where $\arg\max_l \delta_t^l(P)$ is the smallest l maximizing $\delta_t^l(P)$. Let $\Phi(P)$ be the set of packets in P whose laxities do not exceed $h(P)$. We refer to the integer $h(P)$ as the *Hajek-Seri cut* for P.

Considering the set P_t of packets pending in the buffer at time t, the integer $\delta_t^l(P_t)$ is a lower bound on how many packets in P_t must miss their deadlines by time $t+l$ in any schedule of future transmissions. Moreover, if P_t is t-saturated and at time t we serve a packet not in $\Phi(P_t)$ (i.e., one whose laxity exceeds $h(P_t)$), then the number of packets in P_t that will definitely miss their deadlines in the future will increase. Indeed, Hajek and Seri show that scheduling within $\Phi(P_t)$ is necessary and sufficient for throughput optimality.

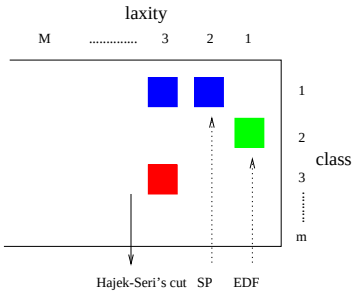

Figure 3: An example where EDF serves more weight than the policy using SP within $\Phi(P_t)$.

Theorem 1 *(Hajek-Seri) A policy π is a TO policy if and only if for any nonempty set of packets P, $\pi_1(P) \in \Phi(P)$.*

Within the class of TO policies, we have considerable freedom in choosing which packet in $\Phi(P_t)$ to serve at each time t. The earliest-deadline-first (EDF) policy provides such a choice. However, if we are interested in minimizing weighted loss, we should choose a packet in $\Phi(P_t)$ based on its class. A naive class-based choice is to select a packet within $\Phi(P_t)$ using SP (i.e., select a highest-weight packet). Such a policy is TO and appears always to serve more important packets than EDF does. Unfortunately, this simple choice does not in fact dominate EDF. Indeed, consider the example given in Figure 3. If there are no further arrivals, EDF serves more weight than π for this traffic. In the next section, we explore a class of policies that make class-sensitive selections within $\Phi(P_t)$ with provable benefits, including dominance of EDF.

We now return to the issue of existence of an OTO policy. The following proposition indicates that, unfortunately, there does not exist such a policy in general.

Proposition 2 *No OTO policy exists if and only if there are at least three classes.*

Proof: A simple modification of the proof of Proposition 1 does the job.

If part: Let i, j, and k be three classes with $w_i > w_j > w_k$. Suppose A_1 consists of a class i packet p' with laxity $l_1(p') = 2$, a class j packet p with laxity $l_1(p) = 1$, and two class k packets with laxities 1 and 2. (see Figure 4). If $A_t = \emptyset$ for all $t > 1$, we must serve p at time 1 to dominate all other TO policies over A. On the other hand, if A_2 consists of a class i packet with laxity 1, and $A_t = \emptyset$ for all $t > 2$, we need to serve p' at time 1 to dominate all other TO policies over A. Hence, no optimal TO policy exists. (The key difference between this example and that in Figure 2 is that the presence of the class k packets ensures that all choices of packet to serve at time 1 are consistent with throughput-optimality.)

Only if part: If there is only one class, then any TO policy is OTO optimal. If there two classes, then any TO policy that serves more class 1 packets than any other TO policy is OTO. Hajek and Seri [11] have shown that such a policy exists (called a MOSTO policy in [11]). ∎

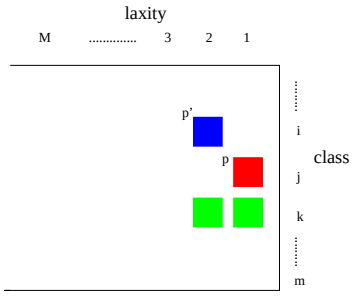

Figure 4: Graphical illustration for the proof of Proposition 2

3 Current-Minloss Scheduling

3.1 Basic description and characterization

In this section, we describe a class of policies called *current-minloss* TO (CMTO) policies. A CMTO policy is a TO multiclass causal scheduling policy that schedules a packet at each time step consistent with the goal of minimizing the weighted loss of the current packets in the buffer. This policy achieves the minimum weighted loss assuming that there are no further arrivals. As we showed in the previous section, the multiclass scheduling problem is unsolvable in general without knowledge of future traffic. Minimizing the weighted loss of packets currently in the queue is in a sense the best we can hope for, without any assumptions on the future traffic (see [3] for some of our work on minimizing weighted loss in the presence of stochastic assumptions on the future traffic). We define and characterize CMTO policies below. In the next section, we give examples of easily implementable CMTO policies.

We begin with some definitions. Let P be a set of packets with deadlines no less than t. A t-schedule of P is a one-to-one partial mapping $i \mapsto p_i \in P$ where $i \in \{1, 2, \ldots\}$ and, if p_i is defined, then $l_t(p_i) \geq 1$. A t-schedule induces a schedule of the packets in P: if p_i is defined, then p_i is scheduled at time $t+i-1$. Note that all packets in a t-schedule are scheduled at times t or later. We can think of a t-schedule as an indexed set of packets $\{p_i : i \in D\}$ where $D = \{i \geq 1 : p_i \text{ is defined}\}$ and $l_t(p_i) \geq i$ for all $i \in D$. A set of packets is said to be *t-schedulable* if there exists a t-schedule that contains the set.

A *maximum-weight t-schedulable subset* of P_t is a set $E_t \subseteq P_t$ such that the total weight of packets in E_t is no less than that of any other t-schedulable subset of P_t. We say that A is an *empty-future* arrival sequence if $A_t = \emptyset$ for all $t > 1$—in such an arrival sequence, all the arrivals happen at time 1. A policy π is a *current-minloss (CM)* policy if, for any empty-future arrival sequence A, we have $L^{\pi}(A) \leq L^{\pi'}(A)$ for any other policy π'. A throughput optimal CM policy is called a *CMTO policy*.

The following theorem characterizes all CMTO policies. We note that the focus on time step 1 exploits the stationarity of the policies we consider to effect all time steps.

Theorem 2 *A policy π is a CMTO policy if and only if for any nonempty set of packets P, there exists a maximum-weight 1-schedulable subset E of P such that $\pi_1(P) \in \Phi(E)$.*

Proof: See Appendix A.1. ∎

Similar to Theorem 2, we can state a characterization of all CM policies—not necessarily TO—involving a different (more relaxed) restriction of packet choice. Specifically, if P_t is not t-saturated, then a CM policy can choose any packet in P_t. However, we will not dwell any further on non-TO CM policies, and will restrict our attention henceforth to CMTO policies.

3.2 Examples of CMTO policies

Theorem 2 suggests the following implementation of any CMTO policy:

For each t, if P_t is not empty, do:

Step 1: Find a maximum-weight t-schedulable subset of P_t (called E_t);

Step 2: Schedule a packet within $\Phi(E_t)$ for service at time t.

We refer to the two-step algorithm above as the *CMTO algorithm*. Next, we provide two examples of CMTO policies based on the CMTO algorithm. These two examples differ only in their implementation of Step 1.

3.3 Step 1 of the CMTO algorithm

Step 1 of the CMTO algorithm takes a given set of packets P_t and computes a maximum-weight t-schedulable subset E_t. These are packets that are "eligible" to be scheduled in the CMTO algorithm; we therefore refer to E_t in Step 1 as the *eligible set*. The computation of E_t from P_t is a familiar problem in "offline" scheduling. Indeed, there are several algorithms described in the literature that can be used for this purpose [14, 17, 16, 18]. Here, we describe two new simple example algorithms for Step 1. We present these two new algorithms not as superior alternatives to existing algorithms, but simply to illustrate what is involved in the calculation of Step 1, and show some of the diversity of approaches that are possible. We note that none of the previous work on such "offline" algorithms proposes to incorporate such methods as part of an online scheduler, as we do (e.g. the algorithms in [16] and [17] were suggested to provide upper bounds on the achievable performance of online scheduling algorithms). We also provide new careful and complete correctness proofs for both our new methods in the appendices.

3.3.1 Forward Algorithm

Our first example algorithm for Step 1 of the CMTO algorithm is called the *Forward Algorithm*, described as follows. We use the notation $P_t(l) = \{p \in P_t : l_t(p) = l\}$ (set of packets in P_t with laxity l), and $M = \max\{l_t(p) : p \in P_t\}$ (largest laxity of packets in P_t).

Forward Algorithm:

1. Input: a set of packets P_t.
2. Initialize: $E(0) = \emptyset$,
3. For $l = 1, \ldots, M$,
   Set $E(l)$ to be the l-most important packets in $P_t(l) \cup E(l-1)$
   (or the whole of $P_t(l) \cup E(l-1)$ if it has fewer than l packets).
4. Output: the set $E_t = E(M)$.

In the Forward Algorithm, we grow an initially empty set into successively larger t-schedulable subsets of P_t until we have reached the largest t-schedulable subset of P_t. At each successive step we keep only the most important (highest-weight) packets that can be scheduled. The worst-case time complexity of the algorithm depends on the data structures used to represent the sets $P_t(l)$ and the current largest t-schedulable subset (as "grown" by the algorithm). One natural approach is to represent these sets with linked lists sorted to be in decreasing order by weight, where packets of identical weight and laxity are combined into a single list entry (we assume the list entries contain a count of the number of packets represented). In this case, the algorithm runs in complexity $O(mM)$, where M is the largest laxity of packets in P_t and m is the number of classes. Note that the key step of computing the set $E(l)$ in the algorithm (step 3) can be implemented as a merge operation running in time complexity $O(m)$ because each list involved will be at most m in length, and this key step will be run at most M times. We note however that if we are to use this representation for the sets $P_t(l)$ during online scheduling, we must be able to efficiently compute P_{t+1} from P_t and A_{t+1}, assuming that P_t is already so represented—when using this representation this computation (essentially sorting A_{t+1}, then merging) has a possibly higher $O(|A_{t+1}| \log |A_{t+1}|)$ cost unless we also assume a similar sorted representation for the new arrivals A_{t+1}. If we assume a fixed bound on both M and m, then this sorting of arrivals can be carried out in $O(|A_{t+1}|)$ time using a bucket sort.

The packets in the output E_t can be listed as a t-schedule (i.e., $E_t = \{p_1, \ldots, p_{|E_t|}\}$ with $l_t(p_i) \geq i$), without increasing the complexity, as long as we choose our representation to order packets of the same weight by laxity (rather than arbitrarily)—this allows the $O(mM)$ conversion of a length M linked list ordered by weight into a length M list ordered by laxity, with the latter being an earliest-deadline-first t-schedule.

We are now ready to prove that the Forward Algorithm does indeed produce a maximum-weight t-schedulable subset of P_t.

Proposition 3 *The Forward Algorithm yields a CMTO policy.*

Proof: See Appendix A.2. ∎

3.3.2 Backward Algorithm

Our second example algorithm for Step 1 of the CMTO algorithm is called the *Backward Algorithm*, described as follows.

Backward Algorithm:

1. Input: a set of packets P_t.
2. Initialize: $p_l = \text{AVAIL}$, $l = 1, \ldots, M$.
3. For $c = 1, \ldots, m$ (from highest to lowest class),
   For packet p in class c, from highest to lowest laxity,
   If $\bar{l} = \max\{j \leq l_t(p) : p_j = \text{AVAIL}\}$ exists, then set $p_{\bar{l}} = p$.
4. Output: the set $E_t = \{p_l : p_l \neq \text{AVAIL}\}$.

The Backward Algorithm can be implemented to run in complexity (essentially) $O(\min(|P_t|, M+m))$, assuming that P_t is represented by giving for each class a linked list of packets sorted by laxity. To see this, note that the algorithm schedules at most M packets and rejects at most m packets (because once a packet is rejected in a given class, the algorithm can move on to the next class). The complexity bound of $O(M + m)$ exploits the Union/Find disjoint-sets algorithm [5], which enables the Backward Algorithm to schedule or reject each packet in essentially constant time (the bound above omits an inverse Ackerman's function factor from the Union/Find algorithm).

A Union/Find implementation can be used to maintain equivalence classes on the laxities, as follows: two laxities are taken to be equivalent if their maximum "schedulable" laxities are the same. Here, the "maximum schedulable laxity" for a given laxity l is the smallest "available" laxity in the schedule p_l not greater than l. Specifically, each time the algorithm sets $p_{\bar{l}}$ to be some packet, the equivalence classes for laxities $\bar{l}$ and $\bar{l}+1$ must be merged. The Union/Find algorithm must be implemented to maintain the smallest equivalence class member as the class representative. Given this maintenance of equivalence classes, the selection of $\bar{l}$ in step 3 in the Backward Algorithm can be done with a single call to "Find" the equivalence class representative of $l_t(p)$.

We note again that creating the required sorted representation of P_{t+1} from P_t as arrivals A_{t+1} are processed can be more expensive than the Backward Algorithm itself if A_{t+1} is originally given in an unsorted fashion, and that a natural assumption would be that M and m are bounded so

that an $O(|A_{t+1}|)$ bucket sort can be used for this purpose. We believe that the algorithms of [16] and [17] can be specialized to similar assumptions as well to achieve similar runtime complexity, though that work does not discuss this specialization and instead focuses on achieving a looser $O(|P| \log |P|)$ bound for an unsorted set of packets P.

Proposition 4 *The Backward Algorithm yields a CMTO policy.*

Proof: See Appendix A.3 ∎

3.3.3 Consistent selection of eligible packets

We say that a CMTO policy is *consistent* if whenever $p \in A_t$ and $p \notin E_{t+k}$ for some $k \geq 0$, then $p \notin E_{t+k+i}$ for all $i \geq 0$. In other words, a consistent CMTO policy has the property that a packet can only be eligible at a time t (i.e., in E_t) if it has been eligible at all previous times since its arrival. It is straightforward to show that if we use either the Forward Algorithm or Backward Algorithm in Step 1 of the CMTO algorithm, then the resulting policy is consistent. Note that in a consistent CMTO policy, if a packet does not join E_t at any time step t after its arrival, we can simply *drop* the packet from the buffer at that time without affecting the future schedule generated by the algorithm, regardless of future arrivals; we will have more to say about such "dropping" policies in Section 5.1.

3.4 Step 2 of the CMTO algorithm

Given a t-schedule $\{p_1, \ldots, p_{|E_t|}\}$ of the eligible set E_t from Step 1 of the CMTO algorithm (as can be obtained from the Forward or Backward Algorithm), Step 2 of the CMTO algorithm can be computed using the following algorithm.

Φ Algorithm

1. Input: $E_t = \{p_1, \ldots, p_{|E_t|}\}$ with $l_t(p_i) \geq i$.
2. Initialize: $i = 1$, $l = l_t(p_1)$, $m = l_t(p_1)$.
3. While $i < |E_t|$ and $i \neq l$,
   $i = i + 1$; $\quad l = \max\{l, l_t(p_i)\}$; $\quad m = \min\{m, l_t(p_i)\}$.
4. If $i = l$ $\quad$ (*i.e. E_t is t-saturated*),
   then set $\Phi_t = \{p_1, \ldots, p_i\}$ $\quad$ (*i is the Hajek-Seri cut for E_t*);
   else set $\Phi_t = \{p \in E_t : l_t(p) = m\}$.
5. Output: Φ_t.

The Φ Algorithm runs in complexity $O(|E_t|)$. Because $|E_t| \leq M$, where M is the largest laxity of packets in P_t, the entire CMTO algorithm can be implemented in $O(M)$. Although Hajek and Seri do not provide an explicit algorithm in [11] for calculating the set $\Phi(E_t)$, they do note that the set can be computed in complexity $O(M)$ if the input is sorted according to laxities. The Φ Algorithm above requires only an input in the form of a t-schedule, not necessarily sorted according to laxities. (In addition, any t-schedulable set sorted by laxities is a t-schedule, and is thus suitable for input to the Φ Algorithm).

Proposition 5 *Given a t-schedule of the set of packets E_t, the Φ Algorithm generates the set $\Phi(E_t)$.*

Proof: See Appendix A.4. ∎

4 Comparison of CMTO and Multiclass EDF

We now provide an analytical comparison between CMTO policies and the earliest-deadline-first (EDF) policy, illustrating the clear advantage of CMTO over EDF. To be more specific, we assume that the EDF policy breaks ties in favor of higher-class packets. To emphasize that this version of EDF extends the usual EDF by making it class-sensitive, we will call this policy EDF+.

We show the following two results. First, a subclass of CMTO policies, called $\text{CMTO}_{\text{EDF+}}$ policies, dominates EDF+; i.e., any $\text{CMTO}_{\text{EDF+}}$ policy achieves no more weighted loss than EDF+ for *any* traffic. Second, there exist traffic sequences such that any CMTO policy achieves a weighted-loss advantage over EDF+ that is arbitrarily close to the maximum possible achievable advantage.

We first show the easier of the two results above—that any CMTO policy can arbitrarily outperform EDF+ for some traffic sequence. We state our result in terms of the "average weighted throughput," which we define as follows. For any finite arrival sequence A, let $T(A)$ be the maximum number of packets in A that can be served (i.e., the number of packets served by any TO policy). Let $W(S^\pi(A))$ the total weight of packets served by π, and define $\bar{W}^\pi(A) = W(S^\pi(A))/T(A)$ as the *average weighted throughput* of policy π for arrival sequence A. Note that for a TO policy π, $\bar{W}^\pi(A)$ is exactly the average weight of packets served by π.

Recall that we have m classes, with w_1 and w_m being the largest and smallest weights, respectively. It is clear that for any TO policy π and any finite arrival sequence A, $\bar{W}^\pi(A)$ is bounded above by w_1 and bounded below by w_m. Therefore, any TO policy can outperform any other TO policy (in terms of average weighted throughput) by at most $w_1 - w_m$. We show that any CMTO policy comes arbitrarily close to achieving this maximum performance advantage over EDF+ for some traffic sequence.

Proposition 6 *For any $\epsilon > 0$, there exists a finite arrival sequence A such that $\bar{W}^\pi(A) - \bar{W}^{EDF+}(A) \geq w_1 - w_m - \epsilon$ for any CMTO policy π.*

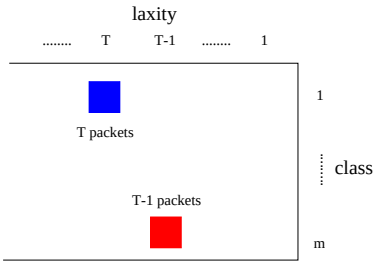

Figure 5: Packets in proof of Proposition 6

Proof: Fix $\epsilon > 0$ and choose $T \geq (w_1 - w_m)/\epsilon$. Let A be an empty-future arrival sequence such that A_1 consists of T class-1 packets with laxity T, and $T-1$ class m packets $p_1, \ldots, p_{T-1}$ with laxity $T-1$; see Figure 5. For this arrival sequence A, any CMTO policy π serves T class-1 packets. In contrast, EDF+ serves $T-1$ class-m packets and one class-1 packet. Hence, $\bar{W}^{\pi}(A) = w_1$, while

$$\bar{W}^{EDF+}(A) = \frac{1}{T}\left((T-1)w_m + w_1\right) \leq \frac{1}{T}\left((T-1)w_m + w_m + T\epsilon\right) = w_m + \epsilon,$$

which completes the proof. ∎

We now show that a particular subclass of CMTO, called $\text{CMTO}_{\text{EDF+}}$, dominates EDF+. This result gives us a traffic-independent preference for $\text{CMTO}_{\text{EDF+}}$ policies over EDF+; i.e., no matter what traffic arrives, EDF+ will not outperform any $\text{CMTO}_{\text{EDF+}}$ policy with respect to weighted throughput.

A $\text{CMTO}_{\text{EDF+}}$ policy is any CMTO policy that uses EDF+ to select the packet to be served from $\Phi(E_t)$ (Step 2 in the CMTO algorithm). Note that a $\text{CMTO}_{\text{EDF+}}$ policy need not actually perform Step 2, because applying EDF+ to $\Phi(E_t)$ is equivalent to applying EDF+ directly to the eligible set E_t, making the calculation of $\Phi(E_t)$ unnecessary. The critical difference between $\text{CMTO}_{\text{EDF+}}$ policies and EDF+ is the inclusion of Step 1, the calculation of E_t at each time step, in $\text{CMTO}_{\text{EDF+}}$.

For convenience, we assume that each packet p has an associated ID number, denoted $ID(p)$, which distinguishes it from all other packets (including those of the same deadline and class). We say that a packet p_1 is *earlier* than p_2 (or that p_2 is *later* than p_1) if one of the following conditions hold:

1. $d(p_1) < d(p_2)$; or
2. $d(p_1) = d(p_2)$ and $C(p_1) < C(p_2)$; or
3. $d(p_1) = d(p_2)$, $C(p_1) = C(p_2)$, and $ID(p_1) < ID(p_2)$.

The above definition induces a total ordering on packets, based primarily on their deadlines. We assume throughout that the EDF+ policy serves the *earliest* packet precisely according to this

ordering. Similarly, at time t, $\text{CMTO}_{\text{EDF+}}$ serves the earliest packet in E_t in the above sense. Note that the earliest packet among a set of packets is unique.

In the remainder of the section, we consider only $\text{CMTO}_{\text{EDF+}}$ policies. Our main result in comparing CMTO policies and EDF+ is the following strong statement on *consistent* $\text{CMTO}_{\text{EDF+}}$ policies.

Theorem 3 *Any consistent $\text{CMTO}_{\text{EDF+}}$ policy dominates EDF+.*

The above theorem establishes that any consistent $\text{CMTO}_{\text{EDF+}}$ policy achieves no more weighted loss than EDF+ for *any* traffic sequence. This result makes a strong statement about consistent $\text{CMTO}_{\text{EDF+}}$ policies relative to EDF+, because these different policies can lead to significantly different queue states over time, and this statement implies that no arrival sequence can drive these different policies to different states so as to favor EDF+ overall. Our proof of this statement is rather involved, requiring special constructs that can be used to compare $\text{CMTO}_{\text{EDF+}}$ policies and EDF+ over time even as their buffer states diverge. In the remainder of this section, we describe the structure of the proof by stating key definitions and lemmas, relegating the rather technical proofs of most lemmas to the appendix.

For the remainder of this discussion, we fix a particular arbitrary finite arrival sequence A and an arbitrary consistent $\text{CMTO}_{\text{EDF+}}$ policy that we will simply call $\text{CMTO}_{\text{EDF+}}$ (a slight abuse of notation). We argue that the policy $\text{CMTO}_{\text{EDF+}}$ achieves no higher weighted loss on A than EDF+. Throughout this discussion, the sets P_t and E_t refer to the evolution of the system under arrivals given by A and service determined by policy $\text{CMTO}_{\text{EDF+}}$.

To prove the theorem, we need to compare the weights of packets served by $\text{CMTO}_{\text{EDF+}}$ and by EDF+. The main issue that complicates this comparison is that the state of the buffer at each time in general will be different for both policies, because as soon as they serve different packets, the evolution of their buffers will take different paths. The key idea in our proof is to identify and characterize *coupling times* in their evolutions—these are points in time when the buffer states are identical for the two policies. It remains then to show that the total weight of packets served by $\text{CMTO}_{\text{EDF+}}$ in between these coupling times is at least that of EDF+.

Lemma 1 *(Time-dominance) At any time while scheduling the arrival sequence A, the packet served by $\text{CMTO}_{\text{EDF+}}$ is no earlier than the packet served by EDF+.*

Proof: See Appendix B.1. ∎

We say that time t is *busy* if P_t (under $\text{CMTO}_{\text{EDF+}}$) is t-saturated. If all times in the interval $[s, t]$ are busy, we say that $[s, t]$ is busy. The basic idea is that we say t is busy when choosing to be idle at time t would necessarily result in additional loss in the future, regardless of further arrivals; in other words, any schedule of P_t that does not serve a packet at time t will serve fewer packets

than a schedule of P_t that serves the maximum number of packets. Note that t is busy if and only if E_t is t-saturated.

The following series of definitions allow us to characterize a particular coupling time that is critical to our proof. We say that d is a *t-saturation time* of the set E_t if $\delta_t^{d-t+1}(E_t) = 0$; i.e., E_t contains exactly $d - t + 1$ packets with deadlines in $[t, d]$. If d is a t-saturation time of E_t, we also say that E_t is t-saturated *at* d. Note that if d is a t-saturation time of E_t or P_t, then $[t, d]$ is busy.

Define the "influence" during interval $[s, t]$ to be

$$I_{[s,t]} = \max\{d(p) : p \in \Phi(E_{t'}), t' \in [s, t]\}.$$

The "influence" $I_{[s,t]}$ is the largest deadline of any packet in the set $\Phi(E_{t'})$ (in Step 2) at any time t' during the interval $[s, t]$. If t is busy, then the laxity $I_{[t,t]} - t + 1$ is simply the Hajek-Seri cut on E_t; i.e., $I_{[t,t]}$ is the smallest t-saturation time of E_t.

Lemma 2 *For busy $[s, t]$, there is a t-saturation time of E_t no less than $I_{[s,t]}$.*

Proof: See Appendix B.2. ∎

Given a busy interval $[s, t]$, Lemma 2 justifies defining $\Sigma_{[s,t]}$ to be the smallest t-saturation time no less than $I_{[s,t]}$. We have the following lemma.

Lemma 3 *(Time-restriction) During a busy interval $[s, t]$, $CMTO_{\mathrm{EDF+}}$ does not serve any packet with deadline exceeding $\Sigma_{[s,t]}$.*

Proof: For any $t' \in [s, t]$, the deadline of any packet in $\Phi(E_{t'})$ does not exceed $\Sigma_{[s,t]}$ (by definition of $\Sigma_{[s,t]}$). But any packet that $\mathrm{CMTO}_{\mathrm{EDF+}}$ serves during the interval $[s, t]$ is an element of such a $\Phi(E_{t'})$. The result follows. ∎

Lemma 4 *For any t, if $\Sigma_{[1,t]} = t$ then $t + 1$ is a coupling time.*

Proof: That $\mathrm{CMTO}_{\mathrm{EDF+}}$ has the desired property follows directly from the time-restriction lemma (Lemma 3). The same property then follows for EDF+ by using the time-dominance lemma (Lemma 1). ∎

We are now ready to identify the coupling times mentioned before, signifying key times when the buffer states for $\mathrm{CMTO}_{\mathrm{EDF+}}$ and for EDF+ are equal. Assume that time 1 is busy (for otherwise, the first busy time can be defined to be time 1 since $\mathrm{CMTO}_{\mathrm{EDF+}}$ and EDF+ perform identically until the first busy time).

Let $c \geq 1$ be such that $c+1$ is the first non-busy time—this time must exist because A is a finite arrival sequence. It follows that $[1, c]$ is a busy interval. We can now show that $\Sigma_{[1,c]} = c$, implying that $c + 1$ is a coupling time. To see this, suppose not for contradiction, i.e. that $\Sigma_{[1,c]} \neq c$. This

implies $\Sigma_{[1,c]} > c$, and thus that E_c has a c-saturation time greater than c. The definition of Σ, the c-schedulability of E_c, and our choice of packet to serve at time c from $\Phi(E_c)$ then imply that $c+1$ is busy, contradicting our choice of c. The coupling at $c+1$ allows us to focus on the interval $[1,c]$ and show dominance there—we can then iterate the same proof by induction on later busy intervals ($\text{CMTO}_{\text{EDF+}}$ and EDF+ perform identically from time $c+1$ until the next busy time). We now focus on showing dominance in the busy interval $[1,c]$.

Given a busy interval $[s,t]$, define

$$\Psi_{[s,t]} = \{p \in E_t : d(p) \leq \Sigma_{[s,t]}\}.$$

The set $\Psi_{[s,t]}$ is always t-saturated (by the definition of $\Sigma_{[s,t]}$) and t-schedulable (by the definition of E_t). Note that we always have $\Phi(E_t) \subseteq \Psi_{[s,t]}$. The following properties of the set $\Psi_{[s,t]}$ allow us to establish the dominance of $\text{CMTO}_{\text{EDF+}}$ over EDF+.

We use the notation $\min(P)$ to denote the smallest weight of any packet in P. Our key lemma asserts that if we fix s, as we increase t the set $\Psi_{[s,t]}$ is in a well-specified sense getting "no less important." Since at each time $t \geq s$ the policy $\text{CMTO}_{\text{EDF+}}$ will be serving a packet in $\Psi_{[s,t]}$, this lemma gives us a useful lower bound on the weight of that packet that will be critical in proving our main theorem.

Lemma 5 *(Ψ-monotonicity) Given a busy interval $[s,t]$, if $\Sigma_{[s,t]} > t$, then* $\min(\Psi_{[s,t]}) \leq \min(\Psi_{[s,t+1]})$.

Proof: See Appendix B.3. ∎

Let $e_1, \ldots, e_c$ be the packets served by EDF+ in $[1, \ldots, c]$, in order, and $m_1, \ldots, m_c$ likewise the packets served by $\text{CMTO}_{\text{EDF+}}$. For any time s in $[1,c]$, let ν_s be the least time $t \geq s$ for which $\Sigma_{[s,t]} = t$. We proved above that $\Sigma_{[1,c]} = c$, which with the definition of Σ implies that ν_s is always in $[1,c]$. Let Λ_s be the number of times in the interval $[s,\nu_s]$ that EDF+ serves a packet never served by $\text{CMTO}_{\text{EDF+}}$. The monotonicity property just shown for Ψ enables us to prove the following result.

Lemma 6 *For any time s in $[1,c]$, every packet served by $\text{CMTO}_{\text{EDF+}}$ in the interval $[s,\nu_s]$ has weight no less than* $\min(\Phi(E_s))$

Proof: The weight $\min(\Phi(E_s))$ is less than or equal to the weight of every packet in $\Psi_{[s,s]}$ and thus less than or equal to the weight of every packet in $\Psi_{[s,t']}$ for any t' such that $s \leq t' \leq \nu_s$ (using Ψ-monotonicity repeatedly). But every packet served by $\text{CMTO}_{\text{EDF+}}$ in the interval $[s,\nu_s]$ is in one such $\Psi_{[s,t']}$ (i.e., every $m_s, \ldots, m_{\nu_s}$ is in its corresponding $\Psi_{[s,s]}, \ldots, \Psi_{[s,\nu_s]}$), so every packet in $m_s, \ldots, m_{\nu_s}$ is as important as $\min(\Phi(E_s))$. ∎

Lemma 7 *For time s such that packet e_s is not in $\{m_1, \ldots, m_c\}$, CMTO$_{\text{EDF+}}$ serves Λ_s packets during $[s, \nu_s]$ never served by EDF+ that are no less important than e_s.*

Proof: The packet e_s cannot be in E_s, or it would be served by CMTO$_{\text{EDF+}}$ at time s and then would be m_s (so that s would not be as assumed). Because e_s is not in E_s, it must be no more important than $\min(\Phi(E_s))$, or E_s would not be the maximum-weight s-schedulable subset of P_s.

The lemma then follows by a simple counting argument given the preceding lemma once we realize EDF+ cannot serve any of $m_s, \ldots, m_{\nu_s}$ outside the interval $[s, \nu_s]$. These packets cannot be served by EDF+ after time ν_s because they will have expired, given the definition of ν_s. They cannot be served by EDF+ at any time t before s because, given the consistency of CMTO$_{\text{EDF+}}$, they must be eligible at any earlier time after their arrival, and thus must be later than the packet served by CMTO$_{\text{EDF+}}$ at time t—but the Lemma 1 states that EDF+ always serves a packet no later than that served by CMTO$_{\text{EDF+}}$. ∎

The above lemma allows an easy proof of the following lemma, completing the proof of Theorem 3.

Lemma 8 *The total weight of packets served by CMTO$_{\text{EDF+}}$ in $[1, c]$ is at least that of the packets served by EDF+ in that interval.*

Proof: Let $i_1 > i_2 > \cdots > i_n$ be the n different times when EDF+ serves a packet e_{i_j} not in $m_1, \ldots, m_c$ (and thus never served by CMTO$_{\text{EDF+}}$). We construct a mapping from the packets e_{i_j} to packets in $m_1, \ldots, m_c$ that are never served by EDF+, such that the mapping always maintains or increases class weight. Note that EDF+ must lose at least m packets among $m_1, \ldots, m_c$ by a counting argument—$m_1, \ldots, m_c$ are c packets with deadlines not exceeding c by the time-decomposition lemma (Lemma 4), and EDF+ spends m times steps in $[1, \ldots, c]$ serving other packets). So our main concern in constructing the mapping above is maintaining or increasing class weight.

For $k = 1$ to n, we simply select a packet p_k served by CMTO$_{\text{EDF+}}$ but not by EDF+ that is no less important than e_{i_k}, ensuring that p_k is not in $\{p_1, \ldots, p_{k-1}\}$ using Lemma 7. The pairs $\{(e_{i_1}, p_1), \ldots, (e_{i_k}, p_k)\}$ constitute a mapping from the packets served by EDF+ but not CMTO$_{\text{EDF+}}$ to those served by CMTO$_{\text{EDF+}}$ but not EDF+ such that the image of any packet e_{i_l} is no less important than e_{i_l}. Because all other packets in $e_1, \ldots, e_c$ are served by both policies, this mapping implies the desired result. ∎

Redefining the first busy time after time c to be time 1, and then selecting a new time c, we can repeatedly apply the above lemmas to new busy intervals (we need apply this argument only a finite number of times because A is a finite arrival sequence). This then establishes that the total weight of packets served by CMTO$_{\text{EDF+}}$ is at least that of EDF+, completing the proof of Theorem 3.

5 Extensions of CMTO Policies

5.1 Scheduling-dropping CMTO policies

Recall that in a consistent CMTO policy, any packet in P_t that is not in the eligible set E_t can be *dropped* from the buffer at time t without affecting the schedule of packets served by the algorithm. This possibility motivates the consideration of policies that decide not only which packets to serve, but also which packets to drop. Such "scheduling-dropping" policies are appealing because of their smaller buffer requirements and their ability to provide more timely feedback on which packets will not be scheduled.

We define a *scheduling-dropping policy* $\bar{\pi}$ as a sequence of pairs $\{(\pi_t, r_t) : t \geq 1\}$, where $\pi = \{\pi_t\}$ is a policy and r_t is a function that maps any nonempty set of packets P_t to a nonempty subset of P_t (representing those packets that $\bar{\pi}$ "retains" in the buffer)—where, as for scheduling policies, we require that the maps involved be stationary, in the same sense as defined there (so that π_1 and r_1 determine π_t and r_t for any t). Given an arrival sequence A and a scheduling-dropping policy $\bar{\pi}$, the sequence P_t of packets in the buffer is given by $P_0 = \emptyset$ and, for all $t \geq 0$,

$$P_{t+1} = A_{t+1} \cup r_t(P_t) - \{p \in r_t(P_t) : d(p) = t \text{ or } p = \pi_t(r_t(P_t))\}\,.$$

We say that a packet p is *dropped* at time t if $p \in P_t - r_t(P_t)$.

The notion of throughput optimality applies similarly to scheduling-dropping policies. It is clear that a TO scheduling-dropping policy maintains a smaller buffer in general than a TO policy that does not drop packets. We say that a scheduling-dropping policy is *TO-buffer-optimal* (TOBO) if the policy is TO and, for any arrival sequence A and any $t \geq 1$, the policy minimizes $|r_t(P_t)|$ over all TO scheduling-dropping policies. In other words, a TOBO policy keeps only those packets that are necessary to preserve throughput optimality.

Hajek and Seri [11] provide the following characterization of TOBO policies, allowing an easy proof that consistent CMTO policies are naturally buffer optimal when extended to drop as described above. Let Q be a subset of P_t. Following the notation of [11], we write $Q \equiv_t P_t$ if the cardinality of the largest t-schedulable subset of P_t is equal to that of Q. We note that the focus on time step 1 exploits the stationarity of the policies we consider to effect all time steps.

Theorem 4 *(Hajek-Seri) A scheduling-dropping policy $\bar{\pi}$ is TOBO if and only if for any nonempty finite set of packets P, we have that $r_1(P) \equiv_1 P$, $r_1(P)$ is 1-schedulable, and $\pi_1(r_1(P)) \in \Phi(r_1(P))$.*

We say that a scheduling-dropping policy $\bar{\pi} = \{(\pi_t, r_t)\}$ is a *CMTO-dropping (CMTOD)* policy if $\pi = \{\pi_t\}$ is a CMTO policy and $r_t(P_t) = E_t$, where E_t is the eligible set associated with π. Any *consistent* CMTO policy can be made into a CMTOD policy without affecting the schedule of packets served, by dropping packets not in E_t at each time t—we also refer to such CMTOD policies as consistent. So, for example, converting a consistent $\text{CMTO}_{\text{EDF+}}$ policy into a CMTOD

policy results in a policy that dominates EDF+, by Theorem 3. The following result establishes that our conversion of a consistent CMTO policy into a CMTOD policy in fact results in buffer optimality.

Proposition 7 *Any consistent CMTOD policy is a TOBO policy.*

Proof: Because E_t is a maximum-weight t-schedulable set, $r_t(P_t) = E_t \equiv_t P_t$ and $r_t(P_t) - E_t$ is t-schedulable. By definition, a CMTOD policy serves a packet in $\Phi(r_t(P_t))$ at each time t. Therefore, by Theorem 4, the desired result holds. ∎

5.2 Incorporation of fair link-sharing in CMTO policies

In this section, we explore the incorporation of "fairness" into CMTO policies. We adopt the following standard framework for fair link-sharing (see, e.g., [10]). Every packet arriving at the queue is associated with one of a finite number of *calls*. We think of these calls as sharing the "bandwidth" of the server (also called the *link*). Each call is targeted to receive a preallocated fraction of the overall bandwidth of the link, expressed as follows. Fix a number T_F representing the interval of time over which we wish to enforce fairness in the link-sharing. The discrete time-line is then divided into intervals of length T_F. Associated with each call i is a number f_i representing the number of time slots that the server should allocate to call i during each interval of length T_F (for convenience, we will use the term "T_F-interval" for such an interval). We call f_i the *link-share allocation* of call i. Naturally, we assume that the link-share allocations for the calls sum to a value less than or equal to T_F. A scheduling policy is considered to provide fair link-sharing if it serves each call approximately according to the prespecified link-share allocations. Note that it is impossible for any causal policy to schedule packets to guarantee the link-share allocations *exactly*, even when such schedules exist. Without explicitly addressing link-sharing fairness, class-sensitive policies—including CMTO policies—can be grossly unfair in the sense that high-class packets can occupy all the server's resources at the expense of starving lower classes of service.

Existing link-sharing schemes in the literature include weighted-fair queueing (WFQ), weighted-round-robin (WRR), and class-based queueing (CBQ) (see [6] and [10]). In [10], Floyd and Jacobson describe a scheme for fair link-sharing based on CBQ. Their scheme considers the class of packets (called "priorities" in [10]), and distributes "excess" time slots according to these priorities. The existing schemes do not explicitly take into account the interplay between multiple classes and deadlines. Here, we describe a scheme to provide fair link-sharing in the presence of both multiple classes and deadlines based on CMTO policies.

Our link-sharing scheme, called CMFQ (CMTO Fair Queueing), is based on a simple idea extending CMTO policies: we consider modifying the CMTO step 1 selection of the "eligible set" with a constrained selection—instead of selecting the maximum weight t-schedulable subset $E_t \subseteq P_t$,

we select the maximum weight t-schedulable subset $\mathcal{E}_t \subseteq P_t$ that can be served without causing any class to exceed its link share allocation in the current T_F-interval. In order to provide "excess bandwidth" control, we then modify this basic algorithm so that if the selected set $\mathcal{E}_t$ can be served entirely without serving any member at the current time, then the current time is deemed excess bandwidth and is scheduled using an unconstrained CMTO policy (without regard to fairness). We now give a pseudocode outline for the class of CMFQ policies, as follows:

CMFQ Algorithm

Whenever $t = kT_F$ for some integer k, set $\mathrm{Cr}_i = f_i$ (initial "credit").

If P_t is not empty, do:

1. Find a maximum-weight t-schedulable subset of P_t (called $\mathcal{E}_t$) such that the number of call-i packets in $\mathcal{E}_t$ does not exceed Cr_i;

2. Let k be the smallest integer such that kT_F is no less than t;
   If $\mathcal{E}_t$ is t-saturated or $|\mathcal{E}_t| \geq kT_F - t + 1$, then

   2a. Schedule a packet p within $\Phi(\mathcal{E}_t)$.

   2b. Set $\mathrm{Cr}_{\mathrm{call}(p)} = \mathrm{Cr}_{\mathrm{call}(p)} - 1$, where call($p$) is the call index of packet p.

   else ("excess bandwidth" exists)

   2c. Find a maximum-weight t-schedulable subset of P_t (called E_t, as usual);

   2d. Schedule a packet within $\Phi(E_t)$.

The CMFQ algorithm schedules packets similar to CMTO but takes into account the "remaining credits" for each class (Cr_i). If there is "excess bandwidth," the algorithm proceeds exactly as in the standard CMTO algorithm. We determine the presence of excess bandwidth by checking whether the "credit-limited" eligible set $\mathcal{E}_t$ can be served within the current T_F-interval even if no member is served at time t. This can be done if and only if both of the conditions checked in line 2 are false: $\mathcal{E}_t$ must not be t-saturated, and the number of packets in $\mathcal{E}_t$ must be small enough to serve by the start of the next T_F-interval without service at the current time.

One natural assumption regarding the relationship between calls and classes is that for every call, all packets of that call belong to the same class. Under this assumption, the computation of $\mathcal{E}_t$ can be accomplished using a simple modification of the Backward Algorithm:

1. Input: P_t, Cr_i for all calls i.

2. Let $\mathcal{P}_t$ be the subset of P_t obtained as follows: for each call i, retain only the Cr_i-largest laxity packets of call i.

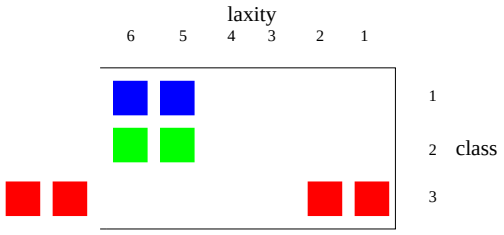

Figure 6: An example that illustrates the weakness of SPFQ.

3. Let $\mathcal{E}_t$ be the result of applying the Backward Algorithm to $\mathcal{P}_t$.

4. Output: $\mathcal{E}_t$.

The correctness of this algorithm rests on the following claim, which the reader can verify: for any "credit-limited" schedule involving packets dropped in step 2, there is an equivalent-weight "credit-limited" schedule not involving such packets (in this schedule, each such packet is replaced by a packet from the same call retained in step 2).

In the Section 6, we evaluate the performance of (an instance of) CMFQ to illustrate its link-sharing fairness. We also compare the policy to the scheme of Floyd and Jacobson [10] as well as a simple credit-based extension of SP, which we call SPFQ. Specifically, SPFQ schedules a packet according to SP among all packets that are in calls with positive credit, where the credits are updated in a similar fashion to CMFQ—the call-i credit value is decremented whenever a call-i packet is scheduled. Our empirical results indicate that while all three schemes above provide fair link-sharing, the weighted-loss incurred by CMFQ policies is significantly smaller than the weighted-losses of the other two schemes.

We conclude this section by describing a simple example that gives insight into the superiority of the CMFQ approach over SPFQ. Consider Figure 6. Currently six packets are in the queues and only two more class-3 packets will arrive in the future. Suppose that fair link-sharing needs to be provided by serving at least two packets of each class over eight time slots (here, classes and calls coincide). On the one hand, any CMFQ policy will serve all of the packets currently in the queue over six time slots, and the incoming class-3 packets over the next two time slots (by scheduling the "excess bandwidth" using CMTO). On the other hand, SPFQ will lose the pending class-3 packets. Even though both policies achieve the goal of fair link-sharing by serving two packets for each class, CMFQ policies incur a smaller weighted loss than SPFQ.

6 Empirical Results

6.1 Weighted-loss comparisons with EDF+ and SP

In this section, we provide quantitative results illustrating the performance of a particular $\text{CMTO}_{\text{EDF+}}$ policy—which uses the backward algorithm for Step 1—in comparison with EDF+ and SP, and with a theoretical lower bound obtained from applying the optimal offline schedule to the arriving packets. For convenience, in the remainder of this section we refer to this particular $\text{CMTO}_{\text{EDF+}}$ policy simply as the $\text{CMTO}_{\text{EDF+}}$ policy. Our results show that the $\text{CMTO}_{\text{EDF+}}$ policy achieves weighted-loss values that are close to the theoretical lower bound, and outperforms EDF+ and SP by up to an order of magnitude.

Our experimental setting was designed to evaluate the above scheduling policies facing practically realistic video traffic with heavy-tailed session durations, under varying overall loads and burstiness. The service rate of the server is adjusted by varying the number of time slots per second—the larger the number of time slots per second, the higher the service rate—by varying the service rate we implicitly vary the burstiness of the traffic, because higher service rates correspond to more aggregation of video calls and consequently smoother traffic for a given load. We show below how performance depends on burstiness by varying the service rate.

We simulate seven classes of video traffic, consisting of video sessions arriving over time with random session durations. At each time slot, each class generates a video session according to a fixed probability, which we call the *session-arrival probability*. The session-arrival probability is the same for all classes, and is varied to set the desired load of the overall traffic—we show below how performance is affected by varying this load. The duration of the sessions follows a Pareto distribution, described below, to simulate heavy-tailed sessions observed in typical network traffic [8]. All packets in a session have the same class. All the packets in a given session have the same initial laxity at arrival, and is set randomly for the session according to a uniform distribution over $[16, 80]$ msec.

Within each session, packets are generated according to a real MPEG video trace. The video trace we used was adapted from an MPEG encoding of the Star Wars movie provided by [2]. Specifically, the trace provided by [2] divides each frame of the Star Wars MPEG into 200-byte packets, and these packets are evenly spaced over approximately 20 milliseconds.

The duration of each session is sampled from a Pareto distribution function $F(x) = 1 - (b/x)^a$, where a and b are fixed parameters (often called the *shape* and *scale* parameters, respectively). We chose the parameters $a = 2$ and $b = 625$ so that the mean duration of each session is 2 sec, reflecting typical sessions found in practice. We selected this duration to model typical internet traffic, giving a 1% chance that a session lasts longer than 10 seconds, so that the overall traffic pattern has noticeable long-range dependence over timescales in minutes. (Note that this pattern of session duration is not chosen to represent typical MPEG videos, but typical internet sessions,

even though our particular sessions are in fact MPEG video sessions, apart from their durations).

We set the weights of the seven classes (1 through 7) such that class i has a weight of ω^{i-1}. By decreasing the parameter ω, we accentuate the disparity in importance between classes, making the scheduling problem more class-sensitive. We show below how performance depends on ω.

We evaluate various online scheduling algorithms relative to the offline optimal weighted loss, a theoretical lower bound on the weighted loss which we computed by applying an optimal offline non-causal scheduling algorithm on the packets involved in the simulation. Specifically, we used the algorithm of [3], although those of [14, 17, 16, 18] would also serve the same purpose. Our results below show that the $\text{CMTO}_{\text{EDF+}}$ policy achieves weighted loss values that are very close to the lower bound, indicating that the $\text{CMTO}_{\text{EDF+}}$ policy is essentially optimal for this particular type of traffic.

We begin exploring the performance of the $\text{CMTO}_{\text{EDF+}}$ policy by measuring the *competitive ratio* achieved by this policy in comparison to that achieved by the simpler EDF+ and SP policies. The "competitive ratio" is the ratio between the performance of an online algorithm and the optimal offline performance for the same traffic. Here, we use weighted loss as the measure of performance, and compute the optimal off-line performance as just described. Figure 7 shows the competitive ratio between the weighted loss achieved by each algorithm and the optimal offline weighted loss (from the theoretical bound mentioned above), as a function of the service rate (number of time slots per second). We vary the service rate while holding the load constant (as described above) to vary the smoothness of the resulting traffic. Each point on the algorithm-evaluation plots represents a simulation of the associated policy over 1,125,000 time slots. For these plots, we set $\omega = 0.6$ and fixed the overall load to be 0.7 (the load is the ratio of the number of packets generated to the total number of time slots), increasing the arrival rate of calls to maintain this load as the service rate increases. For completeness, we also show an alternative perspective on the same data in Figure 7 by showing the weighted loss achieved for the $\text{CMTO}_{\text{EDF+}}$ policy versus that achieved by the EDF+ and SP policies as a function of the service rate, in comparison to the theoretical bound on performance computed by offline computation.

Because the service rate is increasing as we move to the right, the session-arrival rate is also correspondingly increasing, to maintain the fixed load. Therefore, the number of simultaneous sessions in the system is increasing as we move to the right, which corresponds to increasingly smoother traffic. For this reason, we can see that the weighted loss decreases for all policies as the service rate increases, in spite of the fixed load. On the other hand, as the service rate decreases, the variation in traffic over time is more bursty and hence the weighted loss increases.

It is clear from Figure 7 that $\text{CMTO}_{\text{EDF+}}$ outperforms both EDF+ and SP over the entire range of service rates considered, quite substantially at some service rates. At the low end of the range of service rates, the traffic is bursty and involves intermittent periods of heavy traffic loads. Under these circumstances, SP performs well by preferentially serving the highest-weight packets at each

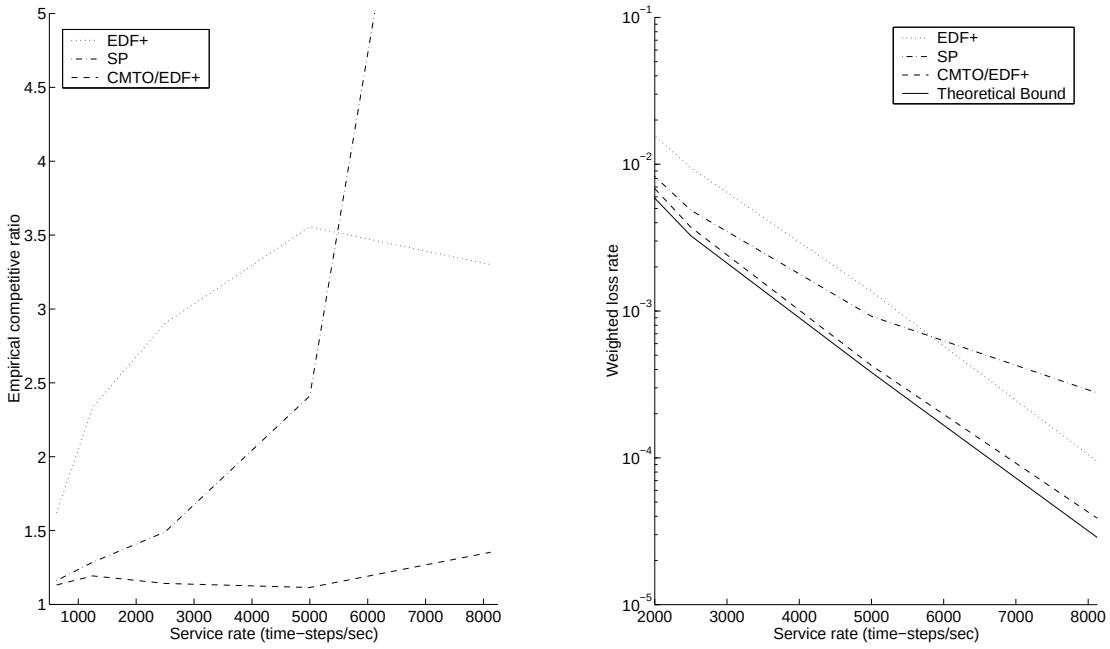

Figure 7: Competitive ratio and log weighted loss over varying service rates with $\omega = 0.6$ and fixed load 0.7. The traffic is getting smoother as we move to the right.

time. On the other hand, at the high end of the range of service rates, the traffic is smooth and therefore throughput-optimal policies such as EDF+ (and $\text{CMTO}_{\text{EDF+}}$) provide the appropriate mechanism for good weighted-loss performance, serving nearly all the traffic. The $\text{CMTO}_{\text{EDF+}}$ policy has a clear weighted-loss advantage over the entire range of service rate values, tailoring its service style automatically to the appropriate traffic condition.

The higher service rates shown in Figure 7 (near and above where SP and EDF+ perform identically) are traffics that we believe show an interesting interplay between class-sensitivity and deadline-sensitivity. For very bursty traffic, the deadline element is less important, as reflected by the reasonable performance of SP on the left extreme of the figure. For very smooth traffic, class becomes less important, at least at this load of 0.7. To further explore the interplay between deadline and class, we choose a fixed arrival rate in this region (7500) and show plots in which we vary either the load (i.e. the session arrival rate) or the multi-class nature of the problem (i.e. the class weight parameter ω).

In Figure 8, we show a plot of the weighted loss of $\text{CMTO}_{\text{EDF+}}$ EDF+, and SP as a function of ω, with a fixed service rate of 7500 and a fixed load of 0.7. For values of ω close to 1, all classes are approximately equivalent in weight, and hence any throughput optimal policy—including EDF+ and $\text{CMTO}_{\text{EDF+}}$—is close to weighted-loss optimal. On the other hand, for values of ω close to 0, only class-1 packets contribute to the weighted-loss, and hence SP approaches optimal performance. We can see that $\text{CMTO}_{\text{EDF+}}$ outperforms both EDF+ and SP over the entire range of ω values,

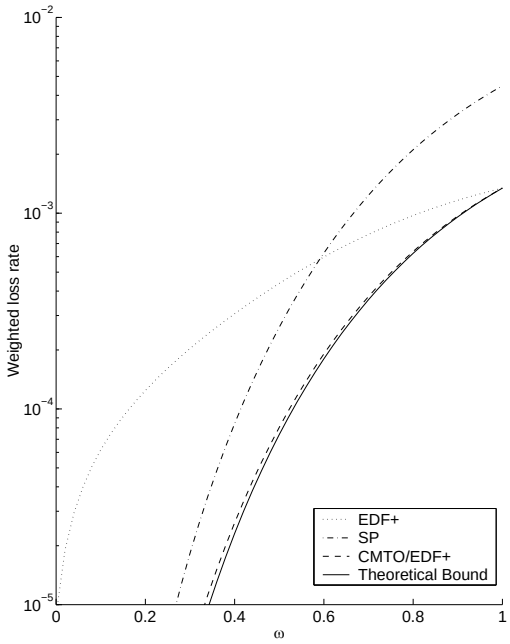

Figure 8: Log weighted loss for varying ω with service rate 7500 and load 0.7.

performing like SP for small values of ω and like EDF+ for large values of ω.

Figure 9 shows a plot of the competitive ratio for $\text{CMTO}_{\text{EDF+}}$ EDF+, and SP as a function of the load, with $\omega = 0.6$ and a fixed service rate of 7500, alongside the alternative weighted loss view of the same data. For these plots, we are using the same aggregated heavy-tailed MPEG trace traffic as for the previous plots, but holding the service rate constant and varying load (rather than fixing load and varying either service rate or ω). At low loads, it is possible to serve almost all packets without missing their deadlines, and hence the throughput optimality of EDF+ and $\text{CMTO}_{\text{EDF+}}$ lead to weighted-loss values close to the optimum (as indicated by weighted-loss values approaching the theoretical bound)—in the figure, we only show loads down to 0.6, where $\text{CMTO}_{\text{EDF+}}$ still shows a substantial advantage over EDF+. As the load increases, the abundance of class-1 packets makes serving only class-1 packets the appropriate way of minimizing the weighted loss. Hence, at high loads, SP significantly outperforms EDF+ and begins to approach the theoretical bound. As we can see, $\text{CMTO}_{\text{EDF+}}$ outperforms both SP and EDF+ over the entire range of load values, indicating the ability of $\text{CMTO}_{\text{EDF+}}$ to tailor its choice of packet to serve based on the load, and to successfully manage the interplay of class and deadline in choosing a packet to serve.

6.2 Buffer occupancy

In Figure 10, we illustrate the straightforward advantage in average buffer occupancy that results from the early dropping of packets in the CMTOD policy (here, $\text{CMTO}_{\text{EDF+}}$ with the dropping extension added) as compared to the EDF+ and SP policies, which do not perform any dropping

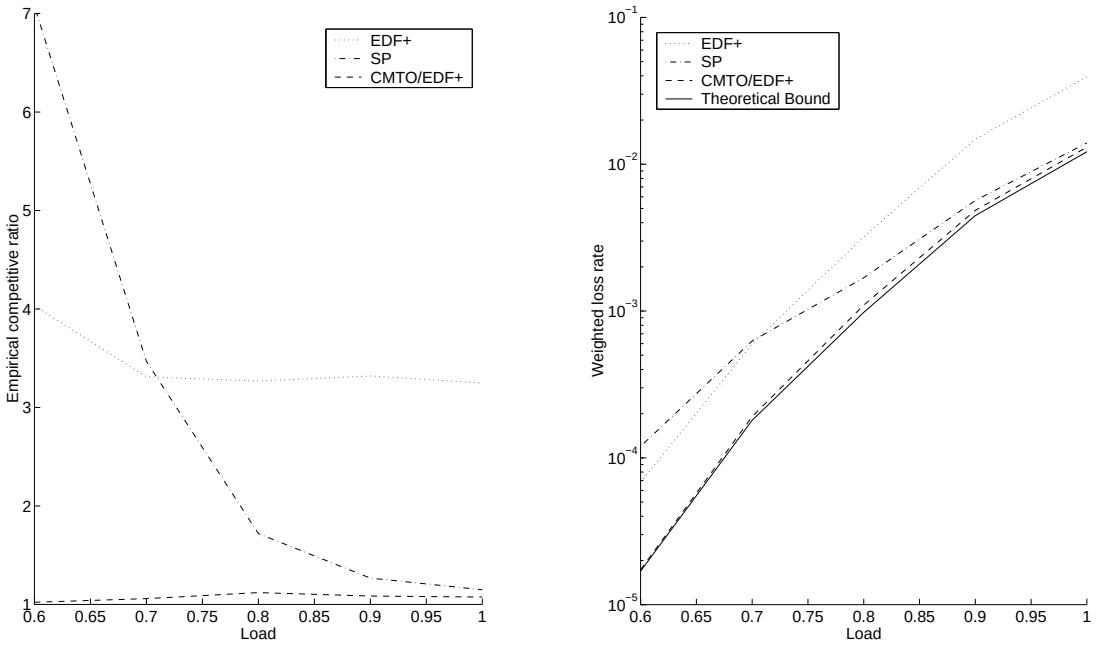

Figure 9: Log weighted loss and competitive ratio for varying load ($\omega = 0.6$, service rate= 7500).

until packets miss their deadlines. We show average buffer occupancy for each policy when faced with the same traffic used for Figure 9 (aggregated heavy-tailed MPEG traffic with the service rate fixed at 7500 and ω fixed at 0.6, with the arrival rate varied to vary the overall load).

The SP policy actually uses less buffer space than CMTOD at low loads, because SP serves far fewer packets than CMTOD by focusing exclusively on the top classes—this focus results in far higher dropping in lower classes, resulting in lower buffer occupancy. But at higher loads, as expected, CMTOD is able to limit the buffer occupancy needed for excellent weighted loss performance, whereas SP and EDF+ both have buffer occupancies that grow severely with growing load. Although we do not explore the issue here, we expect that similar advantages for CMTOD are seen at low loads in highly bursty traffic, which behaves locally like high load traffic during bursts.

6.3 Weighted loss and fairness

Figures 11 and 12 shows a preliminary comparison of the weighted-loss performance of CMFQ relative to SPFQ, CBQ, and CMTO_{SP}. We defer a full comparison to future work—here we test these methods only against very simple traffic rather than using the MPEG aggregation traffic discussed above. Here, we implement a CMFQ policy by using the modified backward algorithm as described in Section 5 for step 1, and SP for step 2 (breaking ties in favor of lower laxity packets). For this experiment, we allocate a 10% link share to each of seven classes (leaving 30% "excess bandwidth"), and take the traffic in each class to be single packets generated with a fixed

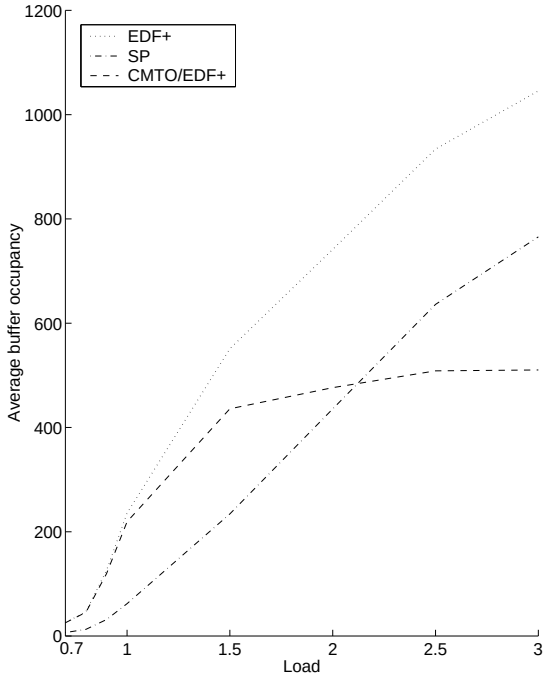

Figure 10: Buffer occupancy levels at service rate 7500.

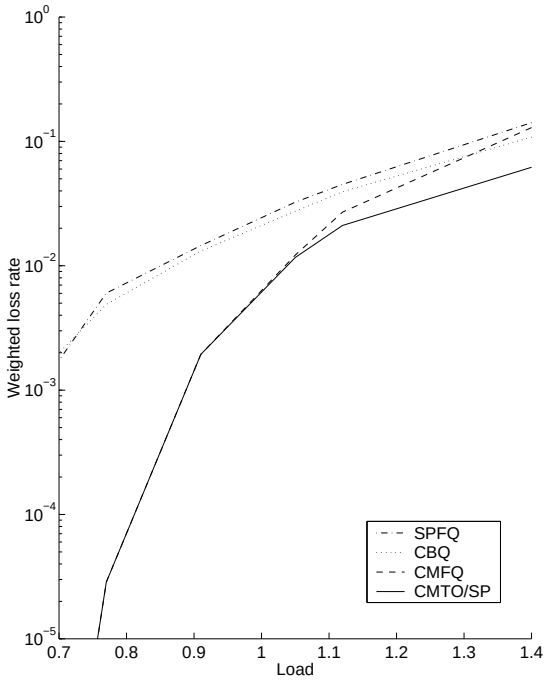

Figure 11: Log weighted loss for link-sharing algorithms and CMTO_{SP} over varying loads.

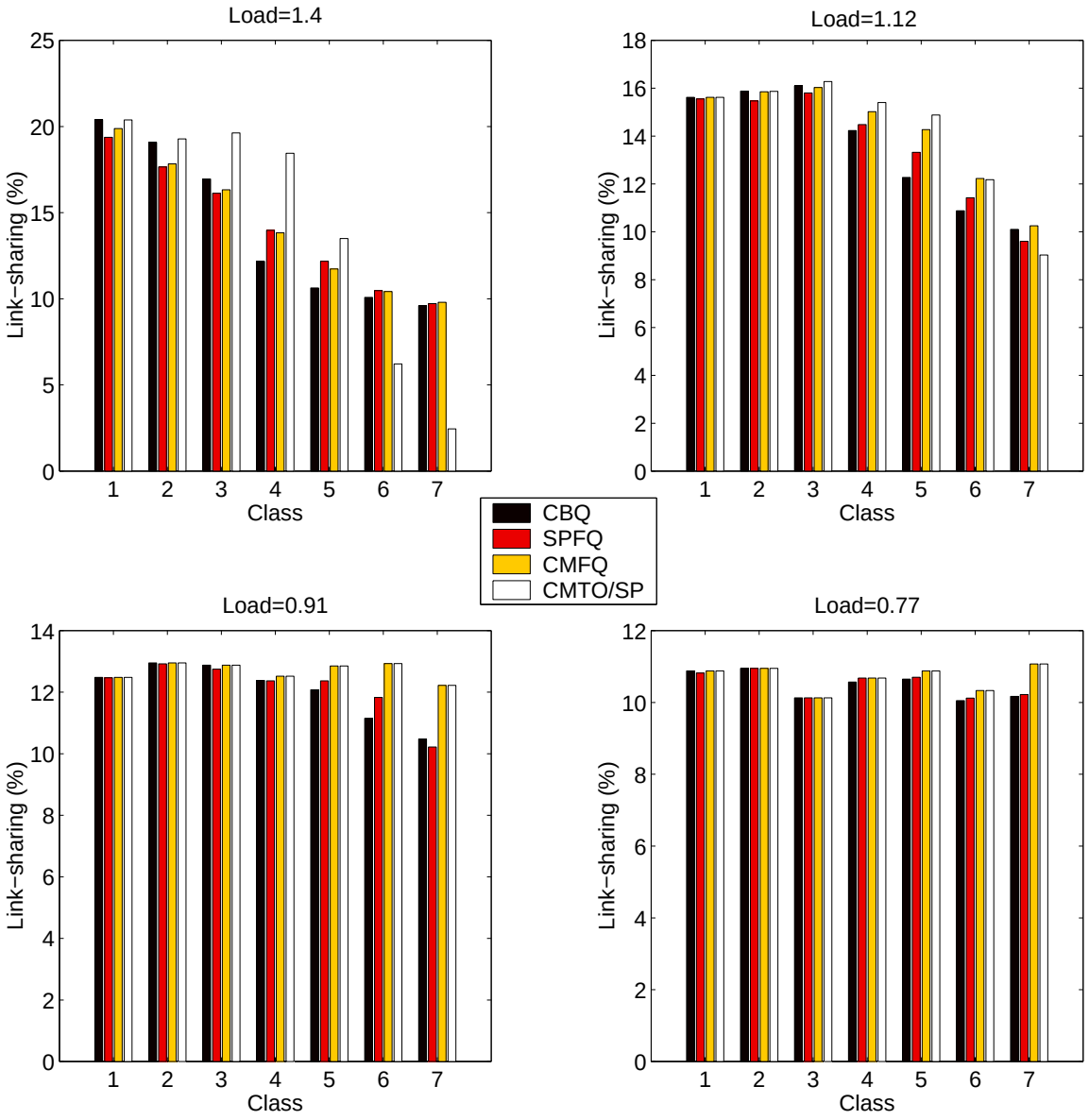

Figure 12: Link-sharing fairness for the same traffic shown in Figure 11 for the same algorithms.

probability at each time step. The probability of packet generation is the same for all seven classes, and is varied to vary the load in the system for the plots shown. CMFQ enforces fairness over a T_F-interval of 600 time steps. The CMTO_{SP} policy shown in the figures uses the backward algorithm for step 1 of the CMTO algorithm, and then static priority for step 2.

At low load values, all policies achieve fairness quite well, with the CM-based policies yielding excess bandwidth benefits to the lowest classes; however, the weighted-loss advantage of CMFQ over SPFQ and CBQ clearly manifests itself with an advantage of orders of magnitude at load 0.7. At low loads, CMFQ behaves like CMTO_{SP} because all classes achieve their desired link-share allocations even in CMTO_{SP}. However, as the load increases, some weighted-loss performance must be sacrificed to provide fairness. At high loads, while CMTO_{SP} achieves a lower weighted loss than the other policies, it does not provide fairness (i.e., it does not maintain the required link-share allocations)—Figure 12 shows that the CMTO_{SP} policy begins to starve the lower classes at load 1.4. On the other hand, CMFQ, SPFQ, and CBQ all continue to provide fairness even at high loads. At loads exceeding 1, the requirement to maintain the link-share allocations dominates the need to minimize the weighted loss, and hence CMFQ performs similarly to SPFQ and CBQ in weighted loss. However, we can see from Figure 11 that at load values close to 1, CMFQ provides superior weighted-loss performance while at the same time preserving fairness as shown in Figure 12. These results show that CMFQ successfully combines the fairness behavior desired in link sharing with the low weighted loss shown by CMTO_{SP}, incurring higher weighted loss only when the link-sharing requirements essentially require such loss.

7 Conclusions

We have shown that no scheduling policy dominates all others over all traffic sequences for our multiclass setting with deadlines. Moreover, the standard EDF+ and SP policies appear to have poor weighted-loss performance over practically reasonable traffic sequences. We have introduced a simple class of CMTO (greedy) policies, which achieves a weighted-loss performance that provably dominates the EDF+ policy. Our examples of CMTO policies illustrate their ease of implementation. These examples—the Forward and Backward Algorithms—also yield consistent policies, which implies that dropping non-eligible packets at each time step does not change the scheduled packets, but reduces the buffer usage to the minimum required to maintain throughput optimality. We have also described a credit-based mechanism based on CMTO to provide fair link-sharing. Our empirical results indicate that a particular instance of a CMTO policy outperforms both EDF+ and SP under practically realistic traffic, with weighted-loss gains of up to an order of magnitude.

We assumed a fixed packet size in this paper. A natural generalization of CMTO policies to the case of variable packet sizes involves assuming that: 1) packet sizes are bounded by deadlines; and 2) the cost incurred for each packet dropped is equal to the packet's length multiplied by its class

weight. In this case, we can simply follow the two-step CMTO algorithm as before—the eligible set (in Step 1 of the CMTO algorithm) can be obtained by applying Sahni's algorithm [18]. However, that characterization of such policies involves extending the definitions of throughput optimality and the current-minloss property.

As we have shown, there is no policy that dominates all other policies over all traffic sequences, unless we have access to some knowledge of future traffic. Access to a probabilistic model of future traffic should aid in scheduling decisions and therefore enable the design of higher-performance schedulers. The analysis of schedulers that incorporate of such a traffic model entails changing the notion of optimality from "traffic-independent" optimality to "probabilistic" optimality (e.g., in terms of the expected total weighted loss over a finite horizon). Preliminary results along these lines are reported in [3].

A Proofs for Section 2

A.1 Proof of Theorem 2

Proof: If part: Let π be a policy and assume that for any nonempty set of packets P, there exists a maximum-weight 1-schedulable subset E of P such that $\pi_1(P) \in \Phi(E)$.

We first show that π is TO. By Theorem 1, it suffices to show that for any nonempty set of packets P, $\pi_1(P) \in \Phi(P)$, i.e., $\pi_1(P)$ has laxity not exceeding $h(P)$. Let P be a nonempty set of packets and E a maximum-weight 1-schedulable subset of P such that $\pi_1(P) \in \Phi(E)$. If P is not 1-saturated, then $E = P$, and so $\pi_1(P) \in \Phi(E) = \Phi(P)$ as desired. So suppose that P is 1-saturated. We show that $h(E) \leq h(P)$ (from which $\Phi(E) \subseteq \Phi(P)$ and thus $\pi_1(P) \in \Phi(P)$ follows immediately). Let $l^* = h(P) = \arg\max_l \delta_1^l(P)$. Hence, for any k, $\delta_1^k(P) \leq \delta_1^{l^*}(P)$. Note that in particular for all $k > l^* \geq 1$,

$$\delta_1^k(P) - \delta_1^{l^*}(P) = N_1^k(P) - k - (N_1^{l^*}(P) - l^*) = (N_1^k(P_1) - N_1^{l^*}(P)) - (k - l^*).$$

and $N_1^k(P) - N_1^{l^*}(P)$ is the number of packets in P with laxities between $l^* + 1$ and k. Similarly,

$$\delta_1^k(E) - \delta_1^{l^*}(E) = (N_1^k(E) - N_1^{l^*}(E)) - (k - l^*).$$

Hence, because $E \subseteq P$,

$$\delta_1^k(E) - \delta_1^{l^*}(E) \leq \delta_1^k(P) - \delta_1^{l^*}(P)$$

for all $k > l^* \geq 1$, from which it follows that

$$\delta_1^k(E) \leq \delta_1^{l^*}(E).$$

Thus, $\arg\max_l \delta_1^l(E) \leq l^*$, which means that $h(E) \leq h(P)$, as desired.

Next, we show that π is a CM policy. Let A be an empty-future arrival sequence, and for each time t let P_t the pending packets at time t on serving A using π. For each t, let E_t be a maximum-weight t-schedulable subset of P_t such that $\pi_t(P_t) \in E_t$. Assume that A_1 is nonempty, for otherwise the result holds trivially. We use the following notation:

- $w_t^* =$ the weight of $p_t^* = \pi_t(P_t)$;
- $W(A_1) =$ total weight of packets in A_1;
- $W(E_t) =$ the total weight of packets in E_t;
- $T =$ the maximum deadline of any packet in A.

We claim that $W(E_{t+1}) = W(E_t) - w_t^*$ for all $1 \leq t < T$. To see this, note that $E_{t+1} \cup \{p_t^*\}$ is a t-schedulable subset of P_t. By definition of E_t, we have $W(E_t) \geq W(E_{t+1}) + w_t^*$, which implies

that $W(E_{t+1}) \leq W(E_t) - w_t^*$. On the other hand, note that $E_t - \{p_t^*\}$ is a $(t+1)$-schedulable subset of P_{t+1} because $p_t^* \in \Phi(E_t)$. Hence, by definition of E_{t+1}, we have $W(E_{t+1}) \geq W(E_t) - w_t^*$.

By the above claim, we deduce that $W(E_T) = W(E_1) - \sum_{t=1}^{T-1} w_t^*$. But $W(E_T) = w_T^*$. Combining the above two equations we obtain $\sum_{t=1}^{T} w_t^* = W(E_1)$. Hence,

$$L^{\pi}(A) = W(A_1) - \sum_{t=1}^{T} w_t^* = W(A_1) - W(E_1).$$

By definition of E_1, we have $W(P_1) - W(E_1) \leq L^{\pi'}(A)$ for any policy π'. Hence, $L^{\pi}(A) \leq L^{\pi'}(A)$ for any policy π', which completes the proof.

Only if part: We use contraposition. Consider a policy π such that for some nonempty set of packets P, $\pi_1(P)$ is not in $\Phi(E)$ for any maximum-weight 1-schedulable subset E of P. Let A be an empty-future arrival sequence. Without loss of generality, we assume that $t = 1$ and that $A_1 = P$ is nonempty.

We consider two cases. First, if P is not 1-saturated, then $E = P$ is a maximum 1-schedulable subset of P. But $\pi_1(P) \notin \Phi(E) = \Phi(P)$ by assumption. Hence, by Theorem 1, π is not TO.

If P is 1-saturated, we claim that the set of packets P' scheduled by π when encountering A do not constitute a maximum-weight 1-schedulable subset of $A = P$, and hence that $L^{\pi}(A)$ is greater than $L^{\pi'}(A)$ for some policy π' that schedules such a maximum-weight subset of P. To see this, suppose not: i.e., that P' is a maximum-weight 1-schedulable subset of P—then because $\pi_1(P) \notin \Phi(P')$ by assumption, there is a packet in $\Phi(P')$ that is not scheduled by π at any time $t > 1$, contradicting the assumption that π schedules all packets in P'. Hence, $L^{\pi}(A)$ is not minimal, and so π is not a CM policy, by definition. ∎

A.2 Proof of Proposition 3

To prove Proposition 3, we will use two lemmas. The first lemma provides the following useful characterization of t-schedulability.

Lemma 9 *A set of packets P (with deadlines no less than t) is t-schedulable if and only if $\delta_t^l(P) \leq 0$ for all $l \geq 1$.*

Proof: If part: We use induction on l to show that if $\delta_t^i(P) \leq 0$ for all $i = 1, \ldots, l$, then the packets in P with laxity not exceeding l can all be served by time $t + N_t^l(P) - 1$. For $l = 1$, if $\delta_t^1(P) \leq 0$, then there is at most 1 packet in P with laxity 1, which (if it exists) can clearly be served at time t. Now assume the desired result for $l - 1$. Suppose that $\delta_t^i(P) \leq 0$ for $i = 1, \ldots, l$. Then, all packets in P with laxity not exceeding $l - 1$ can be served by time $t + N_t^{l-1}(P) - 1$. (by the induction hypothesis). Because $\delta_t^l(P) = N_t^l(P) - l \leq 0$, we deduce that number of packets in P with laxity l is $N_t^l(P) - N_t^{l-1}(P) \leq l - N_t^{l-1}(P)$. But the number of time instances from $t + N_t^{l-1}(P)$ to

$t + N_t^l(P) - 1$ (inclusive) is $N_t^l(P) - N_t^{l-1}(P) \geq l - N_t^{l-1}(P)$, and $t + N_t^l(P) - 1 \leq l$. Hence, all $N_t^l(P) - N_t^{l-1}(P)$ packets with laxity l can be served before their deadlines in the time interval from $t + N_t^{l-1}(P)$ to $t + N_t^l(P) - 1$, completing the induction argument.

Only if part: Suppose $\delta_t^l(P_t) > 0$ for some l. Then, $N_t^l(P) > l$. But in the time interval $\{t, \ldots, t + l - 1\}$, only l packets can be served. Therefore, some packet in P cannot be served by its deadline, which implies that P is not t-schedulable. ∎

The second lemma that we will use to prove Proposition 3 is the following.

Lemma 10 *If E_t is a maximum-weight t-schedulable subset of P_t, and E'_t is any other t-schedulable subset of P_t, then*

(a) $|E_t| \geq |E'_t|$; and

(b) The $|E'_t|$-highest-weight packets in E_t have total weight no less than that of E'_t.

Proof: (a) Suppose $|E_t| < |E'_t|$. Let $k \geq 1$ be the largest integer such that $N_t^{k-1}(E_t) = N_t^{k-1}(E'_t)$ (denote $N_t^0(E_t) = N_t^0(E'_t) = 0$). Thus, $N_t^l(E_t) < N_t^l(E'_t)$ for all $l \geq k$, and so there is a $p \in E'_t - E_t$ with $l_t(p) \geq k$. Also, $\delta_t^l(E_t) < \delta_t^l(E'_t)$ for all $l \geq k$. Because E'_t is t-schedulable, we have $\delta_t^l(E'_t) \leq 0$ (by Lemma 9), and hence $\delta_t^l(E_t) < 0$ for all $l \geq k$. Thus, $\delta_t(E_t \cup \{p\}) \leq 0$ for all $l \geq 1$, which implies (again by Lemma 9) that $E_t \cup \{p\}$ is t-schedulable. Thus, E_t is not a maximum-weight t-schedulable subset of P_t.

(b) We use the notation $w(p)$ for the weight of packet p, and $W(E_t)$ for the total weight of packets in E_t (similarly, $W(E'_t)$, etc.).

Let $n = |E'_t - E_t|$. Because $|E_t - E'_t| = |E'_t - E_t| + |E_t| - |E'_t|$ and $|E_t| \geq |E'_t|$ by part (a), there are at least n packets in $|E_t - E'_t|$. Let D_t be the n-smallest-laxity packets in $E_t - E'_t$, and let $E''_t = (E_t \cap E'_t) \cup D_t$. We have $E''_t \subseteq E_t$ and $|E''_t| = |E'_t|$. We will show that $W(E''_t) \geq W(E'_t)$, which then clearly implies that the $|E'_t|$-highest-weight packets in E_t have total weight no less than that of E'_t.

Note that $W(E''_t) = W(E'_t) - W(E'_t - E_t) + W(D_t)$. Hence, to complete the proof, it suffices to show that $W(D_t) \geq W(E'_t - E_t)$. We use contradiction. Suppose that $W(D_t) < W(E'_t - E_t)$. Let k be the smallest integer in $\{1, \ldots, n\}$ such that the k-smallest-laxity packets in $D_t = E_t - E'_t$ have total weight less than that of the k-smallest-laxity packets in $E'_t - E_t$. Let p be the kth smallest-laxity packet in $D_t = E_t - E'_t$, and p' the kth smallest-laxity packet in $E'_t - E_t$. Then, clearly $w(p) < w(p')$.

We now show that $F_t = (E_t - \{p\}) \cup \{p'\}$ is a t-schedulable subset of P_t. To see this, first consider the case where $l_t(p) \leq l_t(p')$. Thus, for l in the interval $[l_t(p), l_t(p'))$ we have $\delta_t^l(F_t) = \delta_t^l(E_t) - 1 < 0$, and for l outside that interval we have $\delta_t^l(F_t) = \delta_t^l(E_t) \leq 0$. Hence, by Lemma 9, F_t is a t-schedulable. On the other hand, for the case where $l_t(p') < l_t(p)$, we claim that $\delta_t^l(E_t) < 0$ for

l in the interval $[l_t(p), l_t(p'))$. Indeed, for such l we have $N_t^l(E_t) \leq N_t^l(E_t \cap E_t') + (k-1)$. Also, $N_t^l(E_t') \geq N_t^l(E_t' \cap E_t) + k$. But $N_t^l(E_t') \leq l$ by Lemma 9. Thus,

$$N_t^l(E_t) \leq N_t^l(E_t \cap E_t') + k - 1 \leq N_t^l(E_t') - 1 \leq l - 1 < l,$$

proving $\delta_t^l(E_t) < 0$ for such l. Hence, for l in the interval $[l_t(p), l_t(p'))$ we have $\delta_t^l(F_t) = \delta_t^l(E_t)+1 \leq 0$, and for l outside that interval we have $\delta_t^l(F_t) = \delta_t^l(E_t) \leq 0$ as in the other case. Again, by Lemma 9, F_t is a t-schedulable.

Finally, note that

$$W(F_t) = W(E_t) - w(p_i) + w(p_i') > W(E_t),$$

which contradicts the assumption that E_t is a maximum-weight t-schedulable subset of P_t. ∎

We are now ready to prove Proposition 3. We show that the output E_t of the Forward Algorithm is a maximum-weight t-schedulable subset of P_t. For this, we show by induction that for each $l = 1, \ldots, M$, the set $E(l)$ in step 3 is a maximum-weight t-schedulable subset of $P_t(1) \cup \cdots \cup P_t(l)$. For simplicity, write $Q(l) = P_t(1) \cup \cdots \cup P_t(l)$. As before, we use the notation $w(p)$ for the weight of packet p, and $W(E(l))$ for the total weight of packets in $E(l)$ (similarly, $W(Q(l))$, etc.).

Because $E(0) = \emptyset$, $E(1)$ is a highest-weight packet in $P_t(1)$, if one exists, and hence is a maximum-weight t-schedulable subset of $Q(1) = P_t(1)$. Now suppose that $E(l-1)$ is a maximum-weight t-schedulable subset of $P_t(1) \cup \cdots \cup P_t(l-1)$, $l \geq 1$. Consider $E(l)$ and any t-schedulable subset E' of $Q(l)$. It is clear by Lemma 9 that $E(l)$ is t-schedulable. Thus, it remains to show that $W(E(l)) \geq W(E')$.

Write $E' = A' \cup B'$ where $A' = E' \cap P_t(l)$ and $B' = E' \cap Q(l-1)$. By the induction hypothesis and Lemma 10(b), there exists $B'' \subseteq E(l-1)$ such that $|B''| = |B'|$ and $W(B'') \geq W(B')$. Then, $E'' = A' \cup B''$ is a subset of $P_t(l) \cup E(l-1)$, and $|E''| = |A'| + |B''| = |A'| + |B'| = |E'|$. Because $E(l)$ is by definition the l most important packets in $P_t(l) \cup E(l-1)$, it follows immediately that $W(E(l)) \geq W(E'')$, and then that

$$W(E(l)) \geq W(E'') = W(A') + W(B'') \geq W(A') + W(B') = W(E'),$$

which completes the proof. ∎

A.3 Proof of Proposition 4

We show that the output E_t of the Backward Algorithm is a maximum-weight t-schedulable subset of P_t. We use the following notation and terminology. Consider a t-schedule σ; i.e., σ is a partial one-to-one mapping from laxities $\{1, \ldots, M\}$ to packets in P_t such that if $p = \sigma(l)$, then $l_t(p) \geq l$. We say that σ is a *maximum-weight (MW) t-schedule* if the the range of σ is a maximum-weight t-schedulable subset of P_t. We say that σ is *acceptable* if it can be extended to a MW t-schedule. The empty t-schedule is clearly "acceptable" in this sense.

At the start of each iteration of the inner "for" loop in step 3 of the algorithm, the variables p_l define a t-schedule σ where $\sigma(l) = p_l$ if $p_l \neq$ AVAIL; i.e., the range of σ is $\{p_l : p_l \neq \text{AVAIL}\}$. We note that the mapping σ is either extended or not changed at each iteration of step 3, so any packet that cannot be added to the mapping at one iteration will not be addable at any later iteration (we refer below to this property as "schedule extension preserves addability"). We claim that at the start and end of each iteration of step 3, σ is acceptable. We prove this claim by induction. If σ is the empty schedule (which is the case when we start step 3), then σ is acceptable, as noted before.

For the inductive step, we show that if σ is an acceptable t-schedule, and we apply one iteration of step 3, then the resulting t-schedule σ' is also acceptable. Each iteration of step 3 takes a t-schedule σ and extends it to another t-schedule σ' (possibly with no change) as follows. We may assume that σ' is different from σ because it is trivially acceptable otherwise, since σ is. Therefore some packet can be added to σ. Let c be the class of the highest-weight packet that can be added to σ to obtain another t-schedule. Let p be the largest-laxity packet in the class c not already in the range of σ. Then σ' is the t-schedule that extends σ by adding p at the largest laxity $\bar{l}$ such that $l_t(p) \geq \bar{l}$ and $\sigma(\bar{l})$ is undefined (i.e., p_l is AVAIL). In other words, the only difference between σ' and σ, if any, is that $\sigma'(\bar{l}) = p$ whereas $\sigma(\bar{l})$ is undefined. The body of the "for" loops in step 3 computes exactly this $\bar{l}$ (to see this, you must rule out previously rejected packets using the "schedule extension preserves addability" property mentioned above).

To see that the resulting σ' is acceptable, note that some extension σ'' of σ is a MW t-schedule (by definition of acceptability of σ). We now break into two cases. First, if $\sigma''(j) = p$, for some j, then we must have $j \leq \bar{l}$ since $\bar{l}$ was chosen to be the largest laxity available in σ, and σ'' extends σ. Moreover, because p is not scheduled in σ, but is scheduled at $\sigma''(j)$ we can see that $\sigma(j)$ is not defined (and thus $\sigma'(j)$ is not defined unless $j = \bar{l}$). These observations imply that the schedule identical to σ'' except that we swap $\sigma''(j)$ and $\sigma''(\bar{l})$ must be a t-schedule that extends σ'—and since this schedule schedules the same packets as σ'' (which is MW) then it must also be MW, implying as desired that σ' is acceptable. For the second case, we assume that σ'' does not schedule p. Then we can construct a new t-schedule based on σ'' by setting $\sigma''(\bar{l}) = p$. This new t-schedule is clearly an extension of σ'. Moreover, this new t-schedule is also an MW t-schedule, as follows: our choice of p from the highest class c containing packets that can be added to σ, together with the "schedule extension preserves addability" property, ensures that all packets scheduled by σ'' but not by σ have weight at most the weight of p, implying that the packet removed from σ'' by setting $\sigma''(\bar{l}) = p$ has weight no more than p. Hence, σ'' is acceptable, which completes the induction argument.

Finally, note that as step 3 terminates, the computed t-schedule σ must be an MW t-schedule, because by the above claim, σ is acceptable, and by the "schedule extension preserves addability" property there are no more packets that can be used to extend σ (all packets have been considered in step 3 and either added or rejected due to failure of addability). Because the set E_t in step 4 is the range of σ, we conclude that E_t is a maximum-weight t-schedulable subset of P_t. ■

A.4 Proof of Proposition 5

To prove the correctness of the Φ Algorithm, we use the following lemma.

Lemma 11 *Let $E_t = \{p_1, \ldots, p_{|E_t|}\}$ be a set of packets such that $l_t(p_i) \geq i$ for each $i = 1, \ldots, |E_t|$. Then, E_t is t-saturated if and only if there exists $j \in \{1, \ldots, |E_t|\}$ such that $j = \max\{l_t(p_i) : i = 1, \ldots, j\}$.*

Proof: Only if part: If E_t is t-saturated, then there is a j such that $\delta_t^j(E_t) = 0$, i.e., $N_t^j(E_t) = j$. Because $N_t^j(E_t)$ is the number of packets in E_t with laxity not exceeding j, and only packets in $\{p_1, \ldots, p_j\}$ have laxity not exceeding j, we conclude that *all* packets in $\{p_1, \ldots, p_j\}$ have laxity not exceeding j. Hence, $j = \max\{l_t(p_i) : i = 1, \ldots, j\}$.

If part: Suppose $j \neq \max\{l_t(p_i) : i = 1, \ldots, j\}$ for all $j = 1, \ldots, |E_t|$. Because $l_t(p_i) \geq p_i$, for all $j = 1, \ldots, |E_t|$ we have $j < \max\{l_t(p_i) : i = 1, \ldots, j\}$, i.e., there is at least one packet in $\{p_1, \ldots, p_j\}$ with laxity exceeding j. Again because $l_t(p_i) \geq i$ for all i, we deduce that $N_t^j(E_t) < j$, for all $j \in \{1, \ldots, |E_t|\}$. Hence, $\delta_t^j(E_t) = N_t^j(E_t) - j < 0$ for all $j \in \{1, \ldots, |E_t|\}$, and E_t is not t-saturated. ∎

We are now ready to prove the correctness of the Φ Algorithm as stated in Proposition 5.

It is easy to show by induction that each time the algorithm tests the conditions of the "while loop" (in step 3), we have that $l = \max\{l_t(p_k) : k = 1, \ldots, i\}$ and $m = \min\{l_t(p_k) : k = 1, \ldots, i\}$.

We now consider step 4. If E_t is not t-saturated, then by Lemma 11 there can be no j such that $\max\{l_t(p_k) : k = 1, \ldots, j\} = j$, and thus the "while loop" in step 3 cannot terminate with the condition $i = l = \max\{l_t(p_k) : k = 1, \ldots, i\}$ true—as a result, step 4 is reached with $i = |E_t|$ (the only way step 3 can terminate) and $i \neq l$. Hence, the output $\Phi_t = \{p \in E_t : l_t(p) = m\}$ is equal to $\Phi(E_t)$, by its definition, as desired. Now suppose E_t is t-saturated. It is straightforward to check that for any j, $\delta_t^j = 0$ if and only if $j = \max\{l_t(p_k) : k = 1, \ldots, j\}$ (given that $l_t(p_j) \geq j$ for any j). This implies that the loop in step 3 terminates with both i and l equal to the least j such that $\delta_t^j = 0$ (such j exists by Lemma 11 since E_t is t-saturated). But then i is by definition $h(E_t)$, and so the output value $\Phi_t = \{p_1, \ldots, p_i\}$ is by definition $\Phi(E_t)$, as desired. ∎

B Proofs for Section 3

B.1 Proof of Lemma 1 (Time dominance)

We use induction on the time t. For the base case (at time 1), the two policies face the same buffer state, then because EDF+ serves the earliest packet in A_1, $\text{CMTO}_{\text{EDF+}}$ must serve a packet no earlier. For the inductive case, we assume that at all times $k < t$, $\text{CMTO}_{\text{EDF+}}$ serves a packet at time k no earlier than that served by EDF+ at that time. We must show that the packet p_c

served by $\text{CMTO}_{\text{EDF+}}$ at time t is no earlier than that packet p_e served by EDF+. Suppose for contradiction that p_c is earlier than p_e. Then EDF+ must have served p_c at some time $t' < t$. In this case, by the induction hypothesis, $\text{CMTO}_{\text{EDF+}}$ must have served a packet p'_c at time t' such that p_c is earlier than p'_c, and both p'_c and p_c have arrived and are unexpired at time t'. Because $\text{CMTO}_{\text{EDF+}}$ will later serve p_c at time t, by consistency p_c must be in $E_{t'}$. But this contradicts the fact that $\text{CMTO}_{\text{EDF+}}$ uses EDF+ to select the packet served in $E_{t'}$. ■

B.2 Proof of Lemma 2

By definition of $I_{[s,t]}$, there is some $t' \in [s,t]$ and $p' \in E_{t'}$ with deadline d' such that $d' = I_{[s,t]}$ and d' is a t'-saturation time of $E_{t'}$ (because t' is busy and p' is a highest deadline packet in $\Phi(E_{t'})$). If $t' = t$, then we are done. Otherwise, note that because $E_{t'}$ is t'-schedulable and t'-saturated at d', and the packet served at t' is chosen consistent with a schedule of $E_{t'}$, we can choose a subset Δ of $E_{t'} \cap P_{t'+1}$ that is $(t'+1)$-saturated at d' and $(t'+1)$-schedulable such that every packet in Δ expires at or before d'.

Now we show that there must be a $(t'+1)$-saturation time d'' of $E_{t'+1}$ that is at least $d' \geq I_{[s,t]}$. To see this, suppose not, i.e. that there is no such $(t'+1)$-saturation time at d' or later. It follows that there is some packet in Δ that is not in $E_{t'+1}$. Let p'' be a latest expiring such packet (i.e. p'' is a latest expiring packet in $\Delta - E_{t'+1}$). We now have that the set of packets in Δ with deadlines in the interval $[d(p'')+1, d']$ is a subset of $E_{t'+1}$, by our choice of p''. It is straightforward to show by a counting argument that this inclusion, together with the facts that Δ is $(t'+1)$-saturated at d' and $E_{t'+1}$ has been assumed not to be $(t'+1)$-saturated at d', implies that $E_{t'+1}$ is not $(t'+1)$-saturated at any deadline at or after $d(p'')$. But this implies that p'' can be added to $E_{t'+1}$ preserving $(t'+1)$-schedulability, contradicting the maximum schedulability of $E_{t'+1}$. Therefore, by contradiction, there is some $(t'+1)$-saturation time of $E_{t'+1}$ greater or equal to d'.

Replacing t' by $t'+1$, and d' by d'' and repeating the above argument until $t' = t$, we arrive at the desired result by induction. ■

B.3 Proof of Lemma 5 (Ψ-monotonicity)

We prove the lemma by contradiction. Suppose there is a packet p in $\Psi_{[s,t+1]}$ such that the weight of p is strictly less than $\min(\Psi_{[s,t]})$.

We will show that the packet p can be replaced in E_{t+1} by a packet p' from $(P_{t+1} \cap \Psi_{[s,t]}) - E_{t+1}$. The weight of p' must exceed that of p by our assumptions on p since $p' \in \Psi_{[s,t]}$. We show that such p' can be found so that this replacement results in a $(t+1)$-schedulable subset of P_{t+1} of larger total weight than E_{t+1}, contradicting the definition of E_{t+1} as a maximum-weight $(t+1)$-schedulable subset of P_{t+1}.

We first identify the earliest time at which p can occur in any schedule of E_{t+1}. Let d be the

greatest time such that $d < d(p)$ and $\delta_{t+1}^{d-t}(E_{t+1}) = 0$. (We assume by convention that $\delta_{t+1}^{0}(E_{t+1}) = 0$, so that d is well-defined and lies in the interval $[t, d(p) - 1]$.) We note then that $d + 1$ is the earliest time at which p can be scheduled in any schedule of E_{t+1}, and that d is a $(t+1)$-saturation time of E_{t+1} (or is equal to t).

We now argue that the deadline d occurs strictly within the extent of $\Psi_{[s,t]}$, i.e., that $d < \Sigma_{[s,t]}$. Suppose not. We then have that $\Sigma_{[s,t+1]} \geq d(p) > d \geq \Sigma_{[s,t]} > t$. That $d > t$ here implies that d is a $(t+1)$-saturation time of E_{t+1} (rather than being equal to t). However, the definition Σ does not allow $\Sigma_{[s,t+1]} > d \geq \Sigma_{[s,t]}$ with d a $(t+1)$-saturation time of E_{t+1}. This is because $I_{[t+1,t+1]} > d$ contradicts the fact that $I_{[t+1,t+1]}$ is the smallest $(t+1)$-saturation time of E_{t+1}, and $d \geq I_{[t+1,t+1]}$ with $d \geq \Sigma_{[s,t]} \geq I_{[s,t]}$ implies that $d \geq I_{[s,t+1]}$, which contradicts $\Sigma_{[s,t+1]} > d$ (since d is a $(t+1)$-saturation time of E_{t+1}). Therefore, we can conclude by contradiction that $d < \Sigma_{[s,t]}$.

We now show that the desired packet p' exists. We must find a p' in $(P_{t+1} \cap \Psi_{[s,t]}) - E_{t+1}$ such that $(E_{t+1} - p) \cup p'$ is $(t+1)$-schedulable. We ensure schedulability by selecting a p' such that $d(p') > d$. Since there is a schedule of E_{t+1} with p scheduled at $d+1$, we can simply schedule p' at that time in place of p, so long as $d(p') > d$.

We show the existence of p' by contradiction to conclude the proof of Ψ-monotonicity. Suppose for contradiction that:

(*) there is no p' in $(P_{t+1} \cap \Psi_{[s,t]}) - E_{t+1}$ with $d(p') > d$.

Because $\Psi_{[s,t]}$ is t-schedulable (since it is a subset of E_t) and t-saturated at $\Sigma_{[s,t]}$, and a packet p_t was selected for service from $\Phi(E_t)$, we know that $\Psi_{[s,t]} - p_t$ is $(t+1)$-schedulable and $(t+1)$-saturated at $\Sigma_{[s,t]}$. From this we can conclude that there are at least $\Sigma_{[s,t]} - d$ packets in $\Psi_{[s,t]} - p_t$ with deadlines in $[d+1, \Sigma_{[s,t]}]$. Denote the set of these packets Γ. Since $d + 1 \geq t + 1$, $\Gamma \subseteq P_{t+1}$. Then, by our supposition above for contradiction (*), $\Gamma \subseteq E_{t+1}$. But since d is a saturation time of E_{t+1}, there are also $d - t$ (from $d - (t+1) + 1$) packets in E_{t+1} with deadlines in $[t+1, d]$—call the set of these packets Δ. The sets Δ and Γ together ensure that $\Sigma_{[s,t]}$ is a $(t+1)$-saturation time of E_{t+1}. But note that the packet p cannot be in Δ because its deadline is greater than d, and cannot be in Γ because $\Gamma \subseteq \Psi_{[s,t]}$ and the weight of p is less than $\min(\Psi_{[s,t]})$, and Δ and Γ together constitute all the packets in E_{t+1} with deadlines not exceeding $\Sigma_{[s,t]}$. Since $p \in E_{t+1}$, $d(p)$ must thus exceed $\Sigma_{[s,t]}$. But then the fact that $\Sigma_{[s,t]}$ is a $(t+1)$-saturation time of E_{t+1} contradicts the choice of d as the greatest such time less than $d(p)$. This contradiction ensures that the desired p' exists, concluding the proof. ■

References

[1] S. Bajaj, L. Breslau, and S. Shenker, Uniform versus Priority Dropping for Layered Video, ACM Computer Communication Review 28 (1998) 131–143.

[2] L. Breslau, Example Traffic Trace for NS, http://www.research.att.com/breslau/vint/trace.html

[3] H. S. Chang, R. Givan, and E. K. P. Chong, On-line Scheduling via Sampling, in: *Proc. 5th Int. Conf. on Artificial Intelligence Planning and Scheduling*, CO, 2000, pp. 62–71.

[4] R. Cocchi, S. Shenker, D. Estrin, and L. Zhang, Pricing in Computer Networks: Motivation, Formulation, and Example, IEEE/ACM Trans. on Net. 1 no. 6 (1993) 614–627.

[5] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, (MIT Press, 1990).

[6] A. Demers, S. Keshav, and S. Shenker, Analysis and simulation of a fair queuing algorithm, J. of Internetworking: Research and Experience 1 (1990) 3–26.

[7] W. Feng, D. D. Kandlur, D. Saha, and K. G. Shin, Adaptive Packet Marking for Providing Differentiated Services in the Internet, in: *Proc. Int. Conf. on Network Protocols*, 1998, pp. 108–117.

[8] A. Feldmann, Characteristics of TCP connection arrivals, in: *Self-similar Network Traffic and Performance Evaluation*, eds. K. Park and W. Willinger, John Wiley and Sons, 2000, pp. 367–399.

[9] S. Floyd and K. Fall, Promoting the Use of End-to-End Congestion Control in the Internet, IEEE/ACM Trans. on Net. 7, no. 4 (1999) 458–472.

[10] S. Floyd and V. Jacobson, Link-sharing and Resource Management Models for Packet Networks, IEEE/ACM Trans. on Net. 3, no. 4 (1995) 365–386.

[11] B. Hajek and P. Seri, On causal scheduling of multiclass traffic with deadlines, in: *Proc. IEEE Int. Symposium on Information Theory*, Cambridge, MA. 1998, pp. 166.

[12] B. Hajek and P. Seri, Lex-Optimal Multiclass Scheduling with Deadlines, submitted to Mathematics of Operations Research.

[13] E. L. Lawler, *Combinatorial Optimization : Networks and Matroids*, (Holt, Rinehart and Winston, New York, 1976).

[14] E. L. Lawler and J. M. Moore, A Functional Equation and Its Application to Resource Allocation and Sequencing Problems, Management Science 16 (1969) 77–84.

[15] T. L. Ling and N. Shroff, Scheduling Real-Time Traffic in ATM Network, in: *Proc. IEEE INFOCOM*, 1996, pp. 198–205.

[16] J. M. Peha, Heterogeneous-Criteria Scheduling: Minimizing Weighted Number of Tardy Jobs and Weighted Completion Time, J. of Computers and Operations Research 22, no. 10 (1995) 1089–1100.

[17] J. M. Peha and F. A. Tobagi, Evaluating scheduling algorithms for traffic with heterogeneous performance objectives, in: *Proc. IEEE GLOBECOM*, 1990, pp. 21–27.

[18] S. Sahni, Algorithms for Scheduling Independent Tasks, J. of the ACM 23 (1976) 116–127.